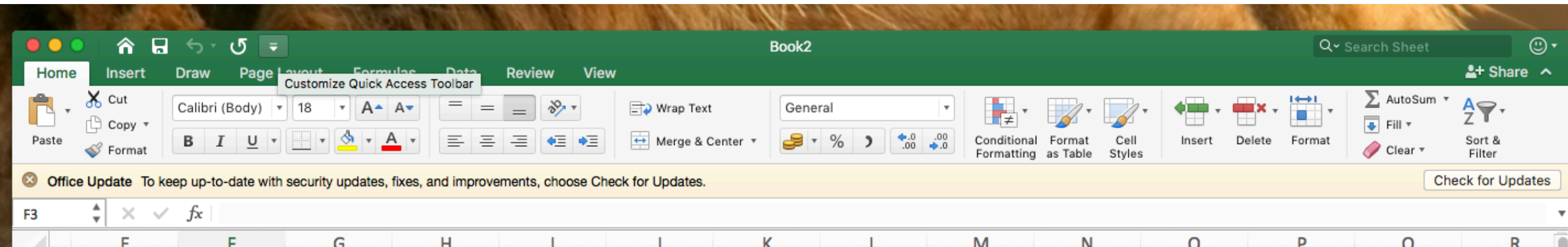


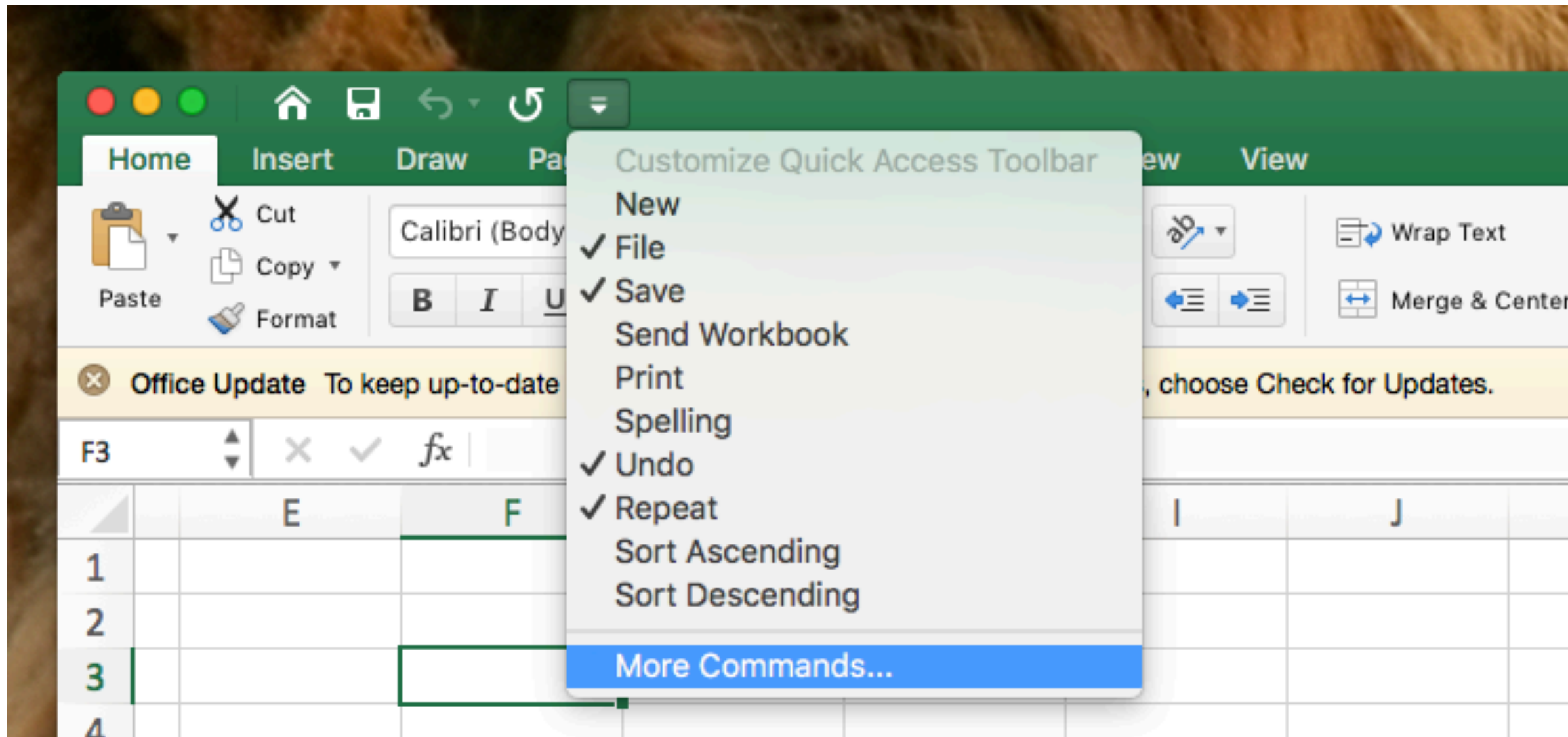
Microsoft Excel

VBA

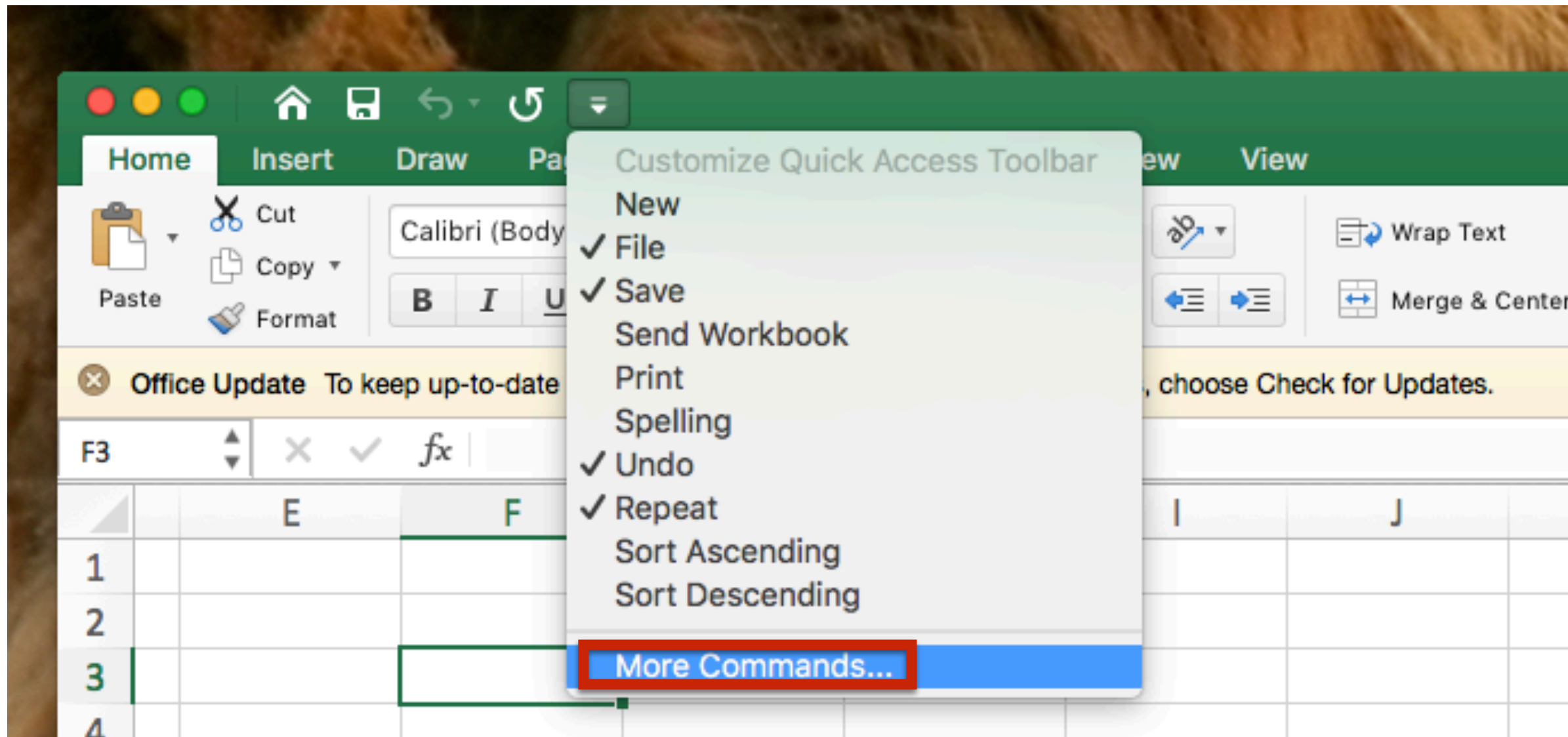
Adding the Developer to the Ribbon

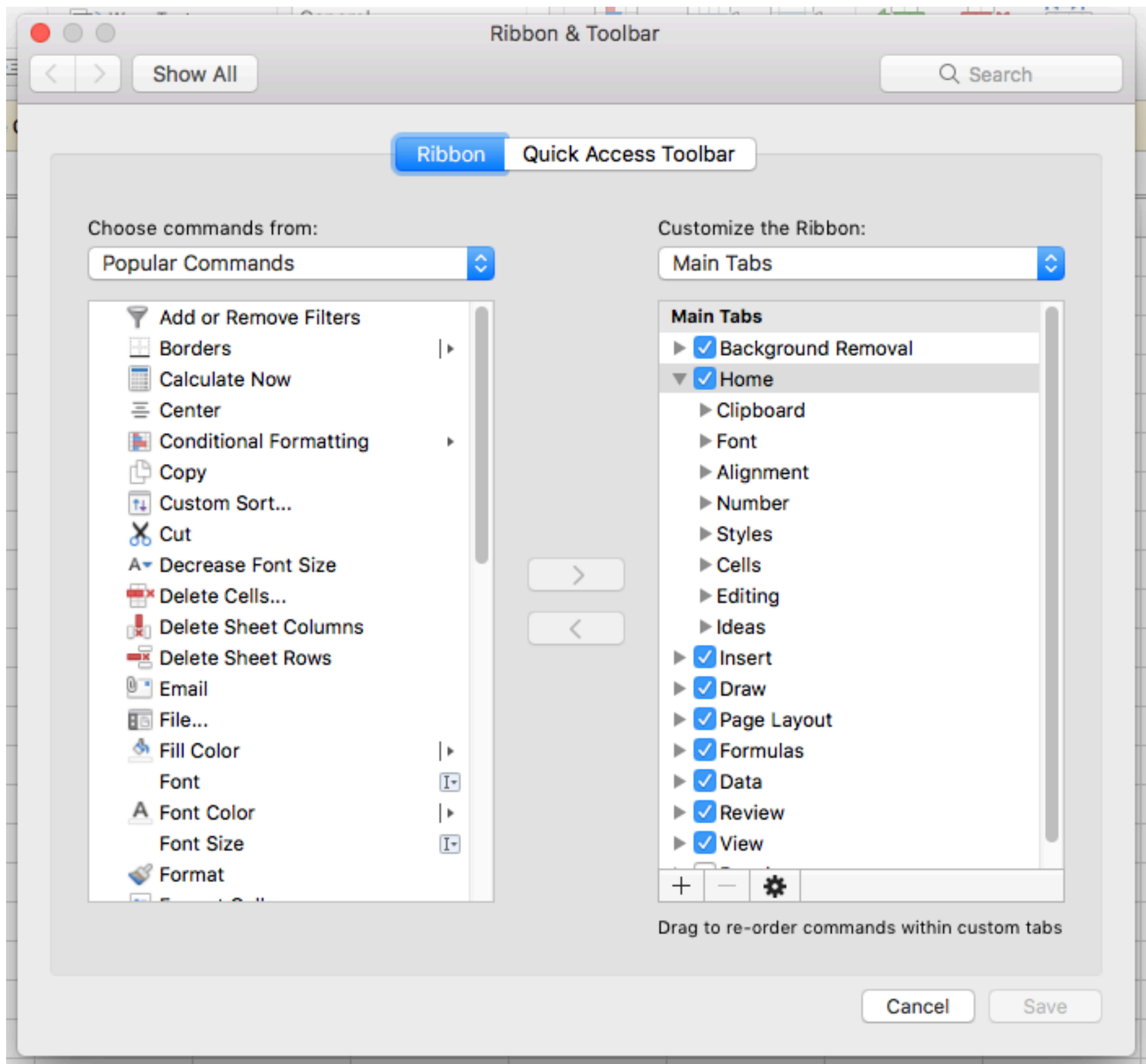


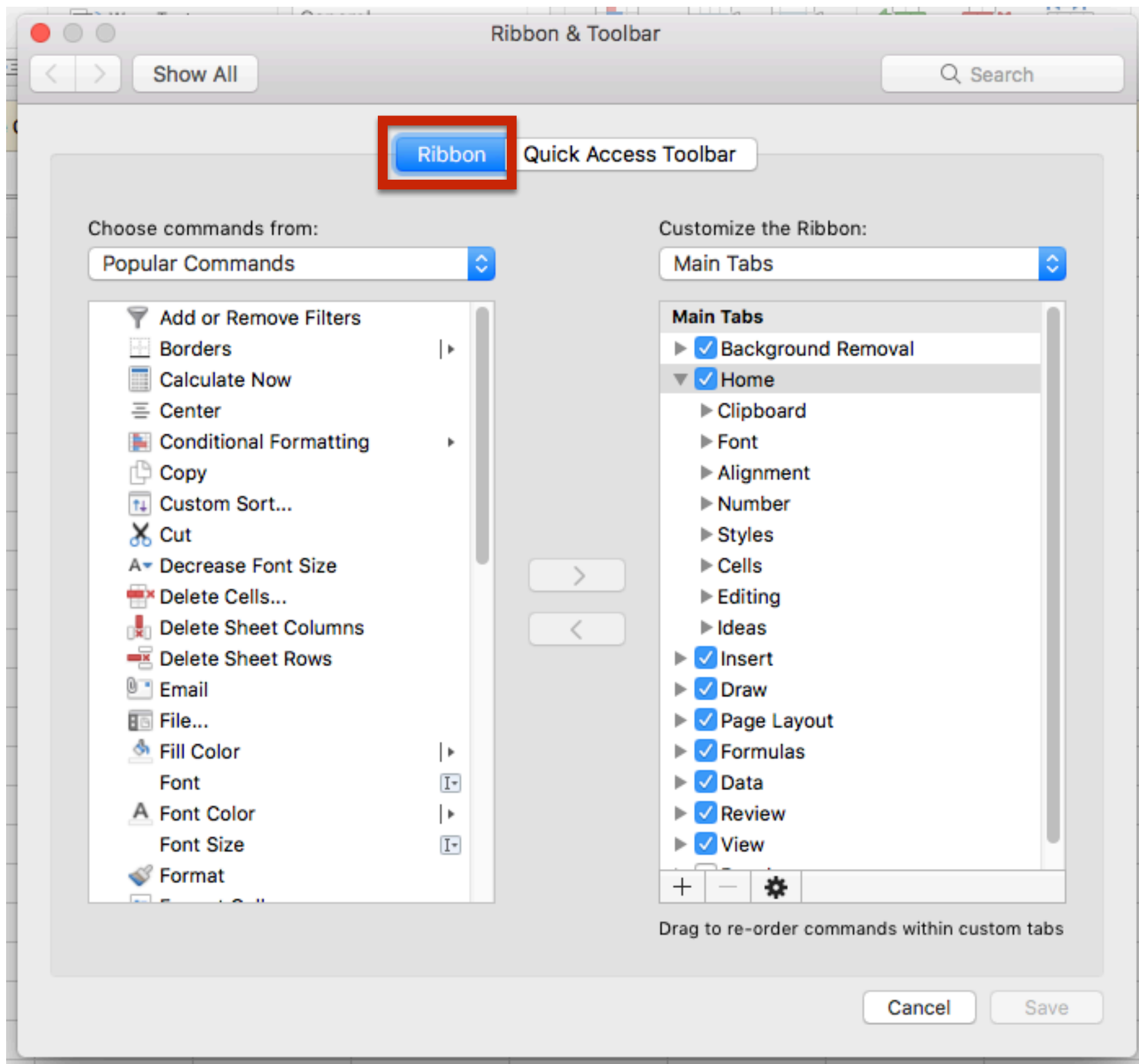
Adding the Developer to the Ribbon

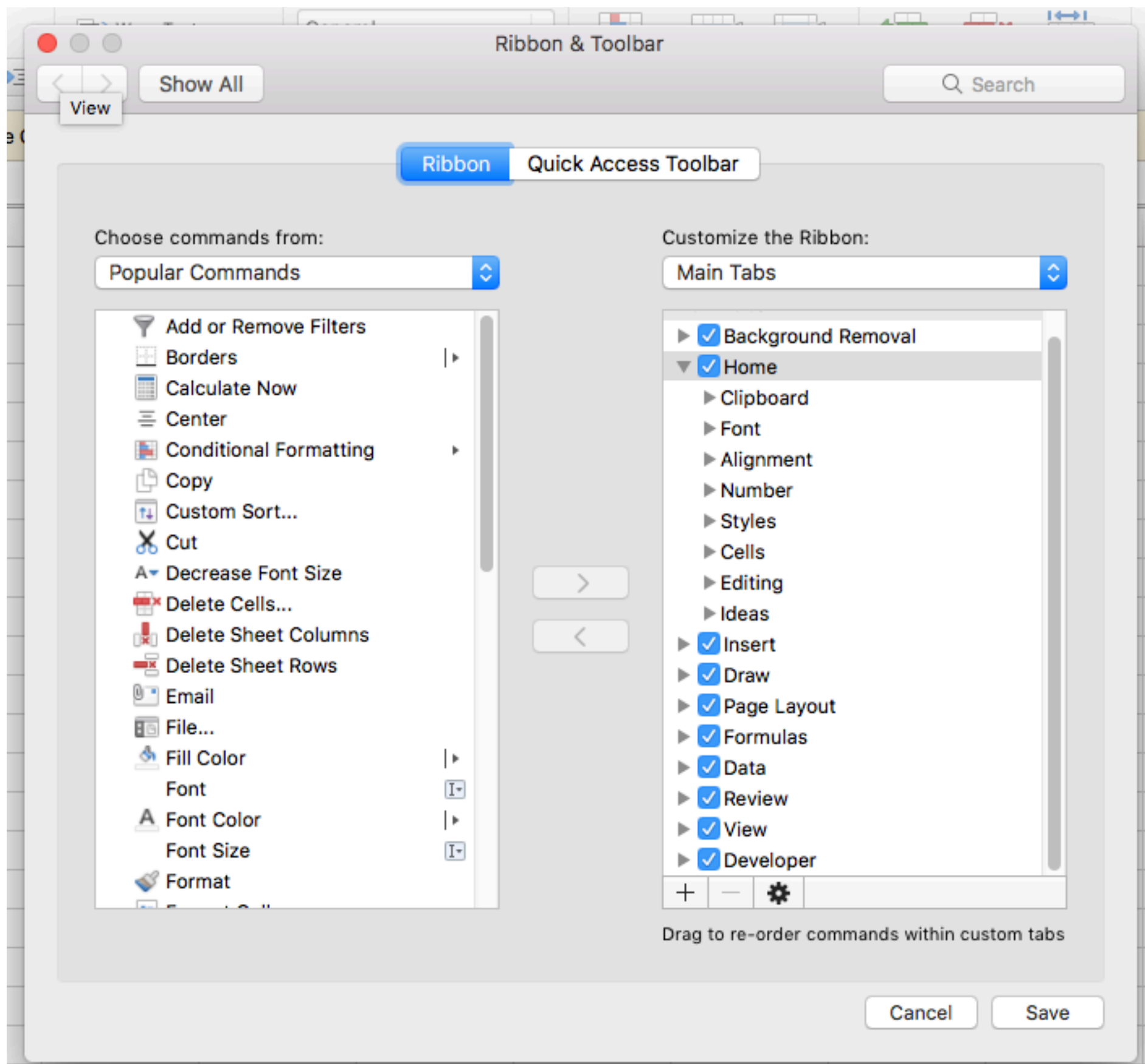


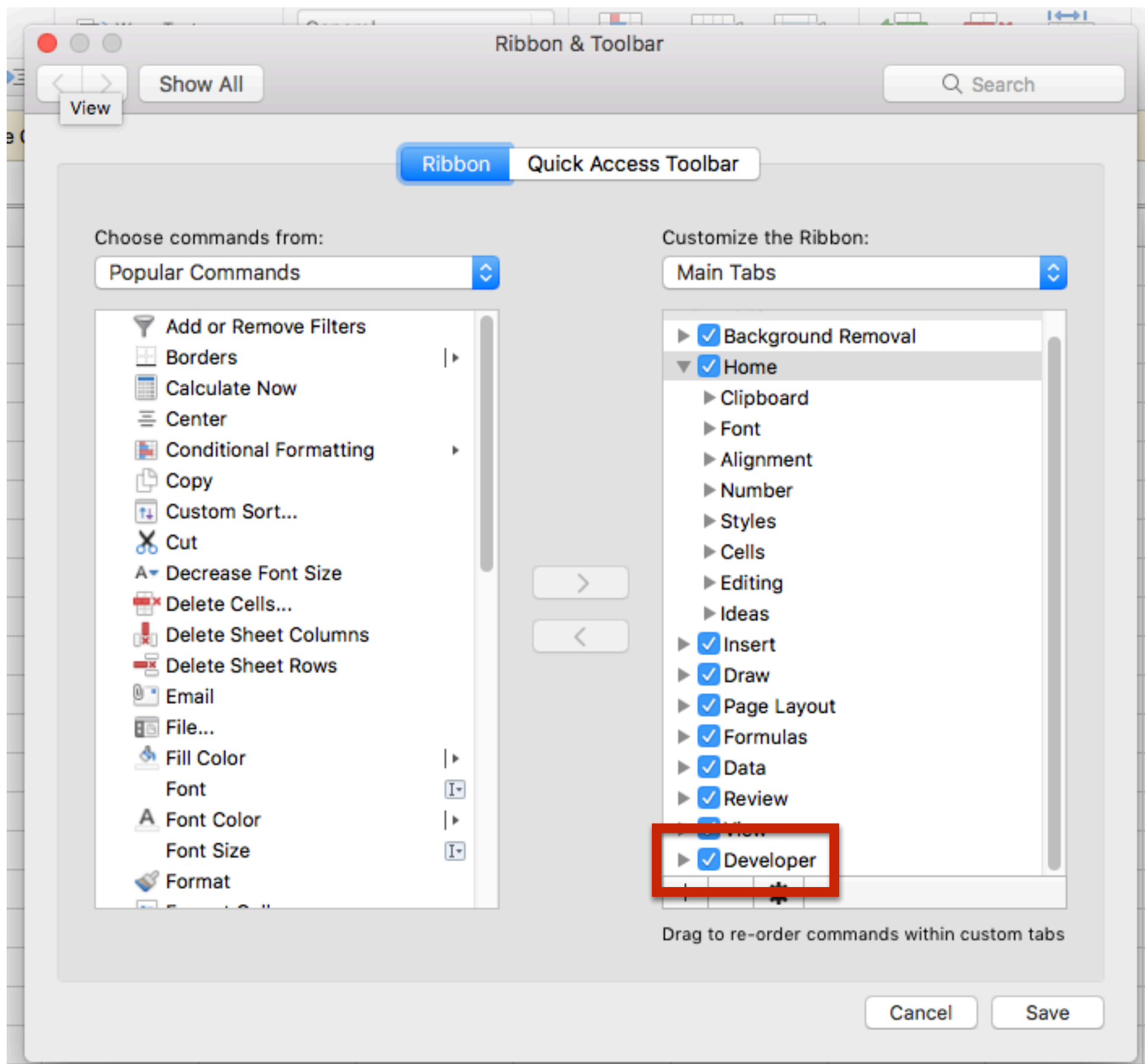
Adding the Developer to the Ribbon



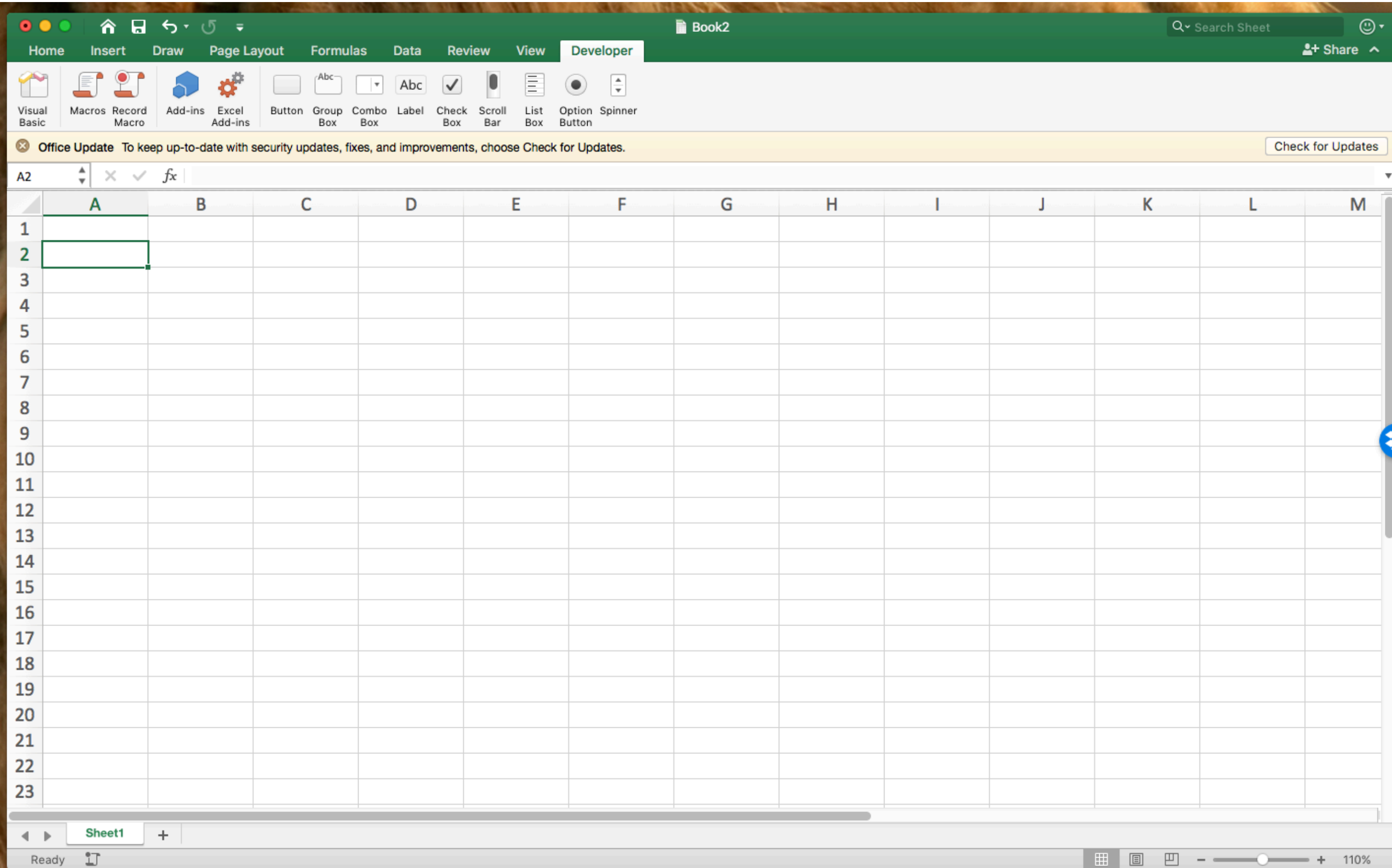




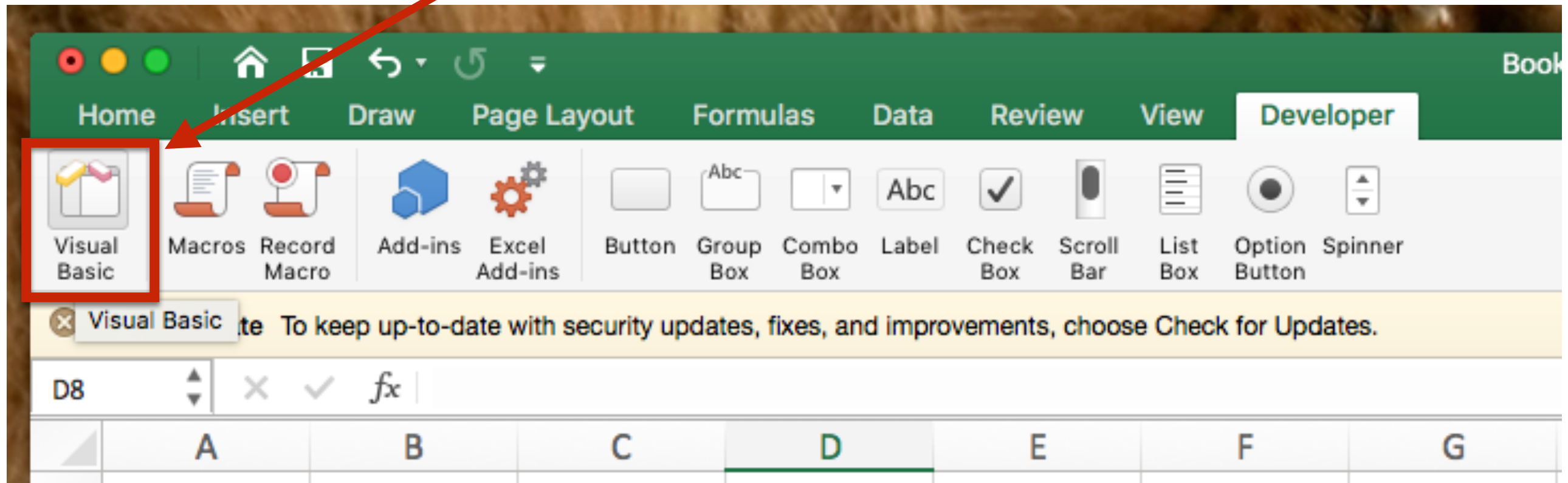




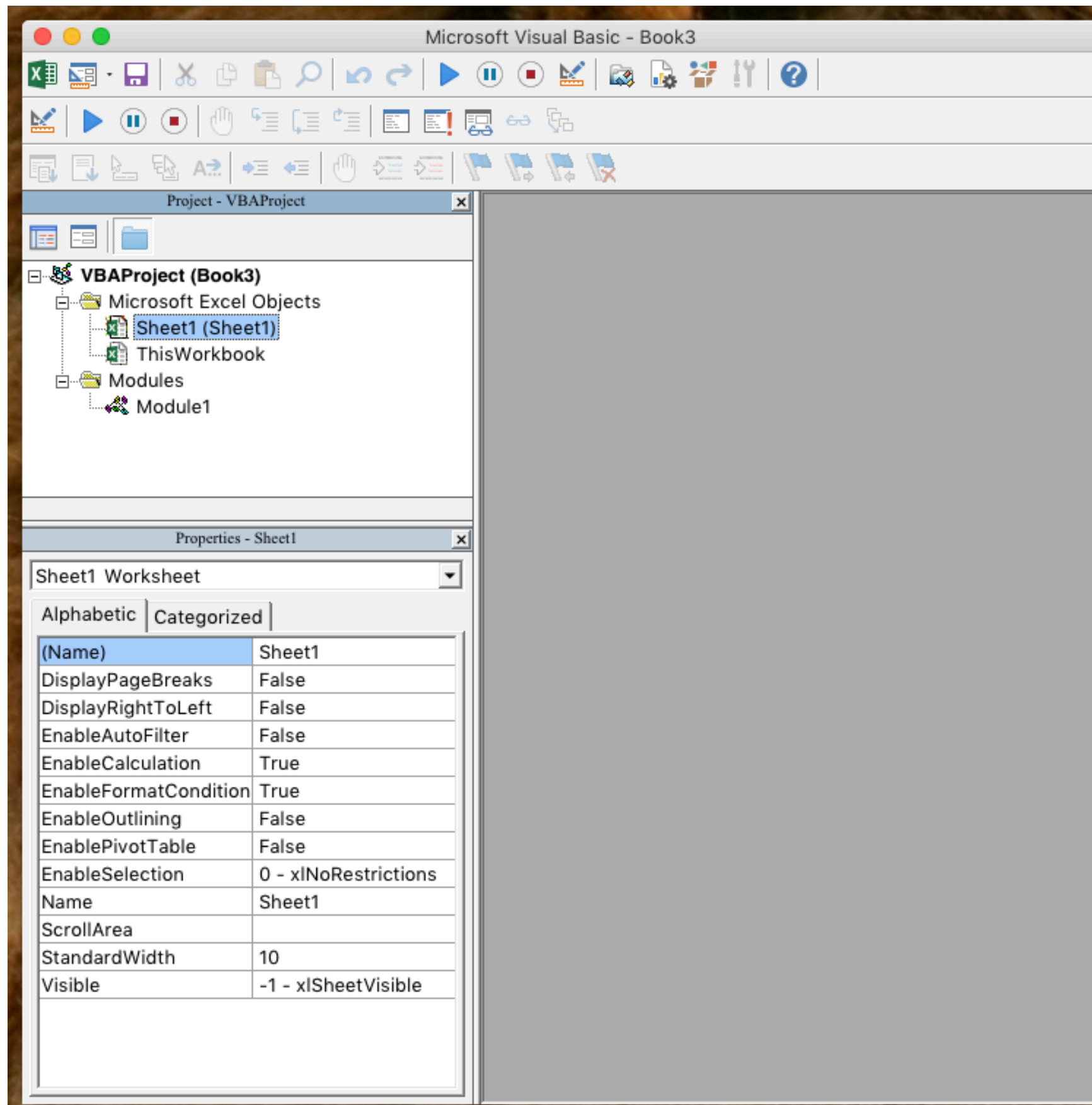
Structure of the code: adding modules



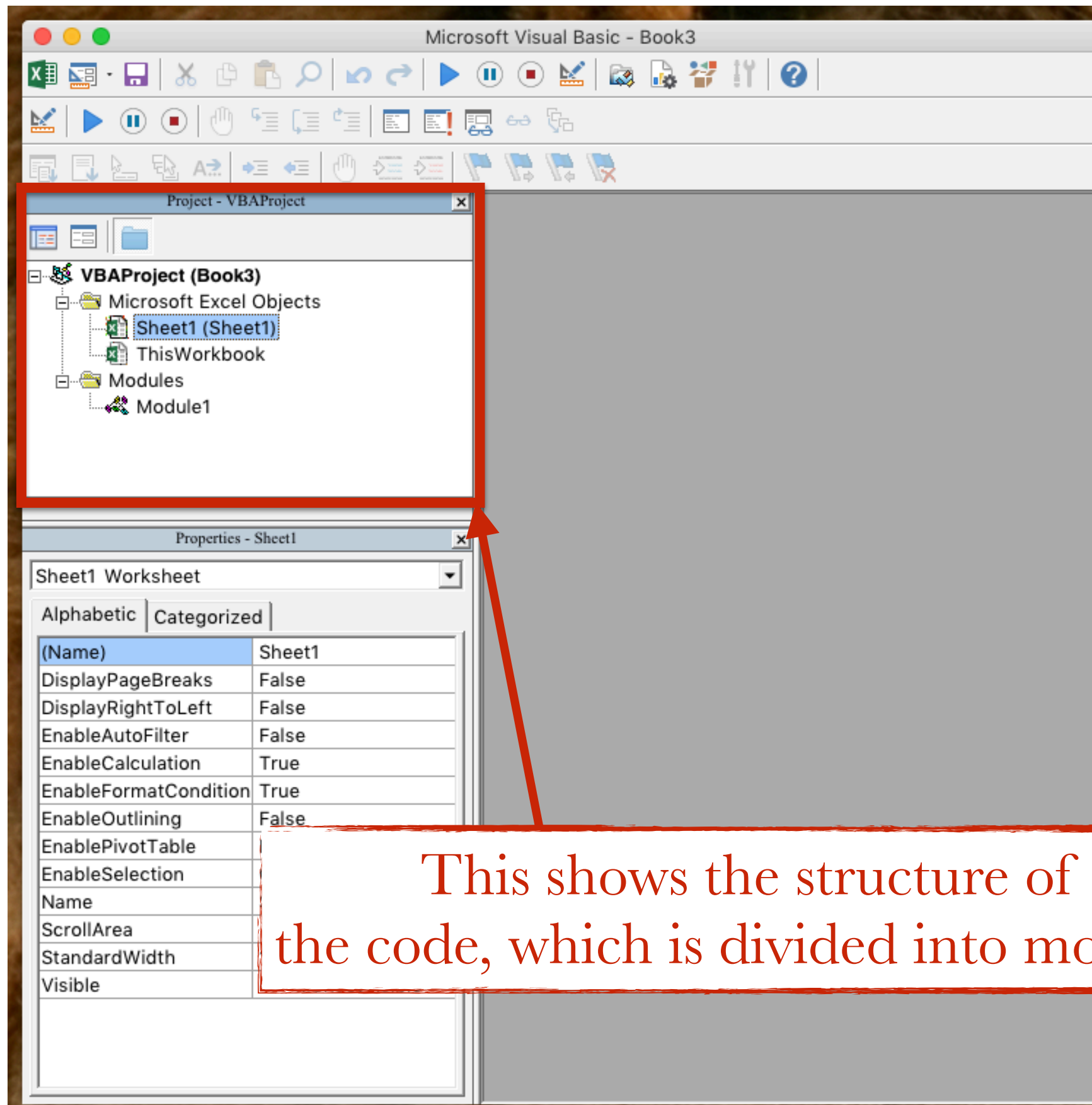
Click to open the VBA editor



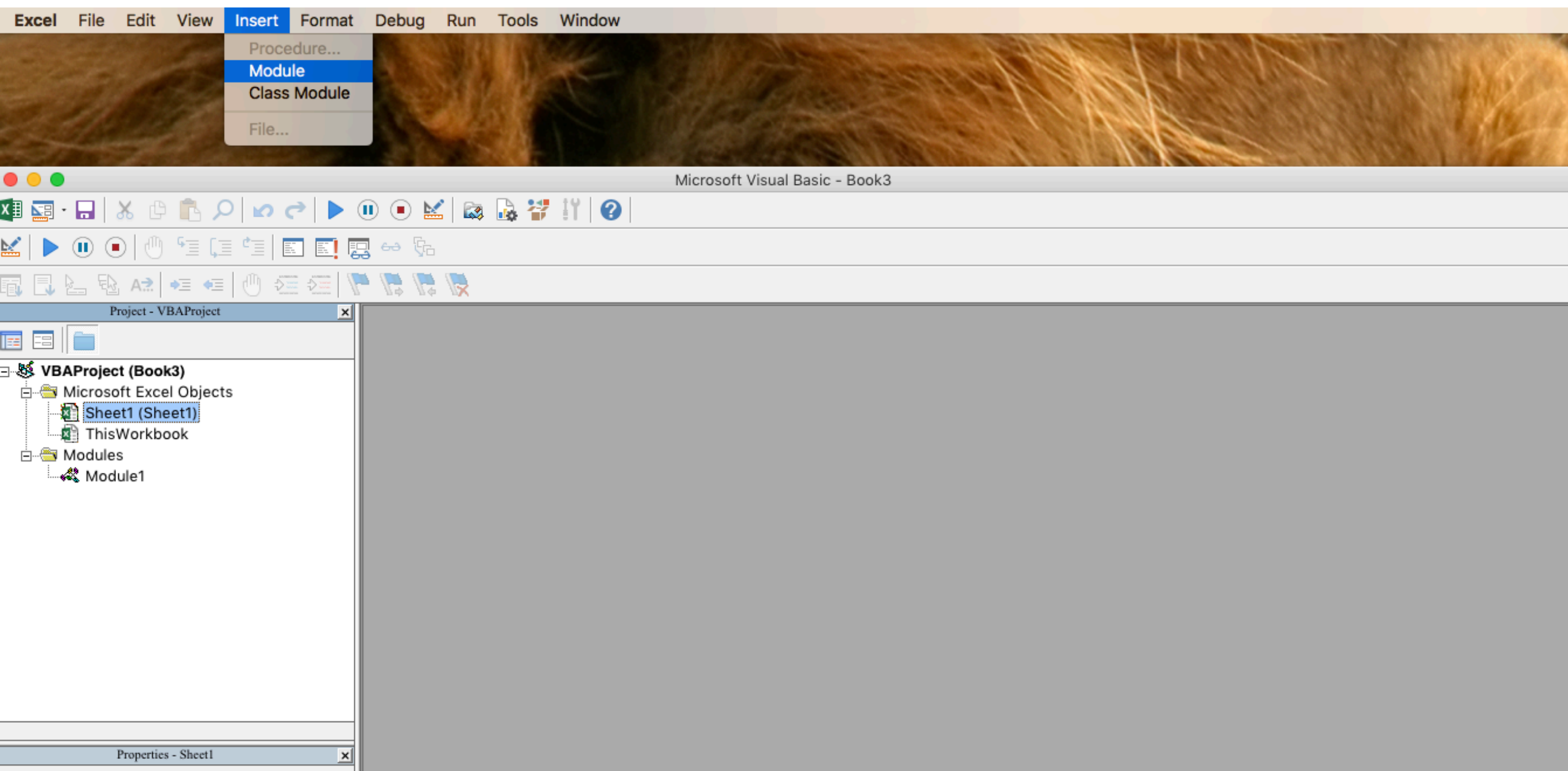
The VBA editor

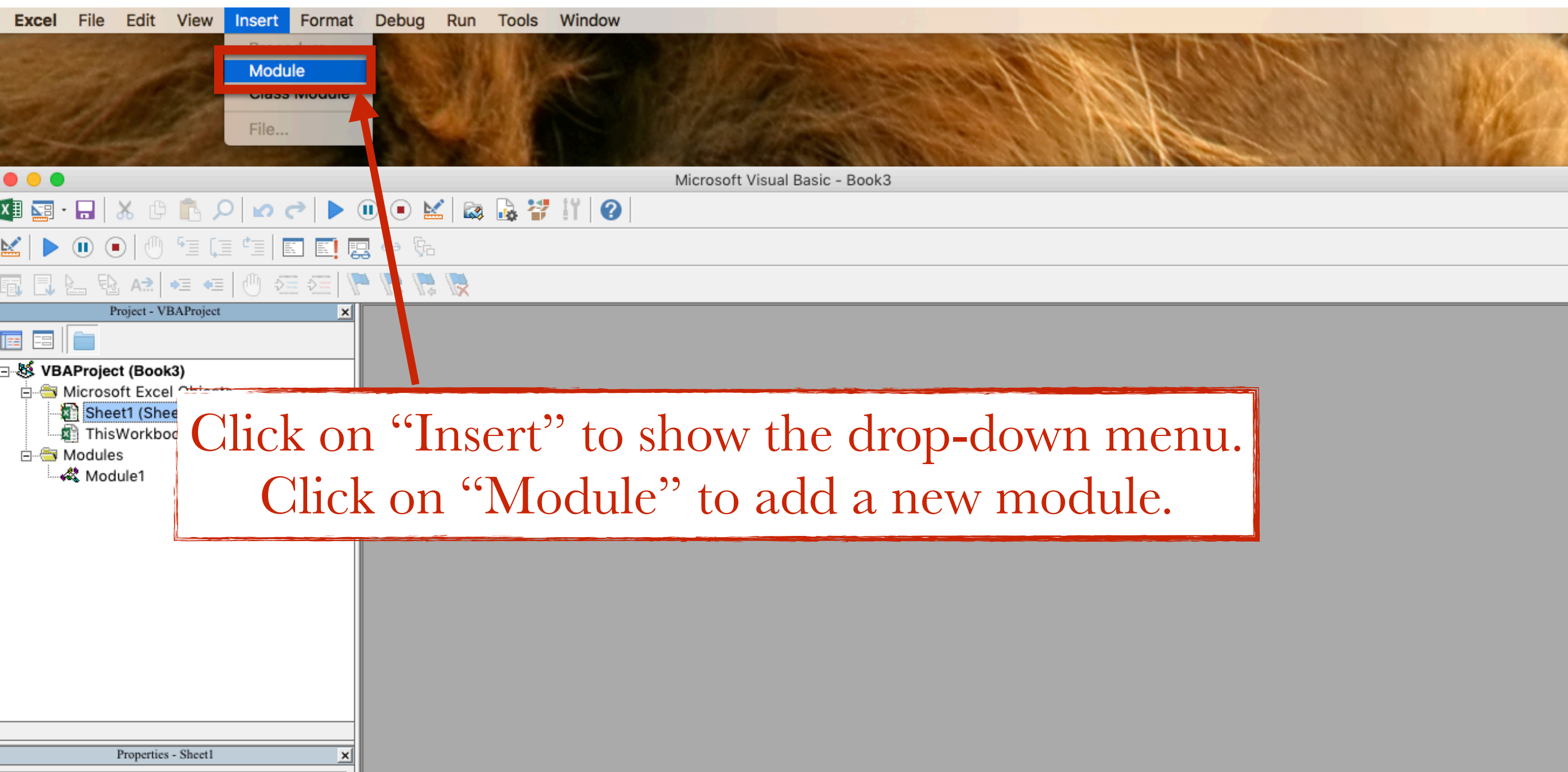


The VBA editor

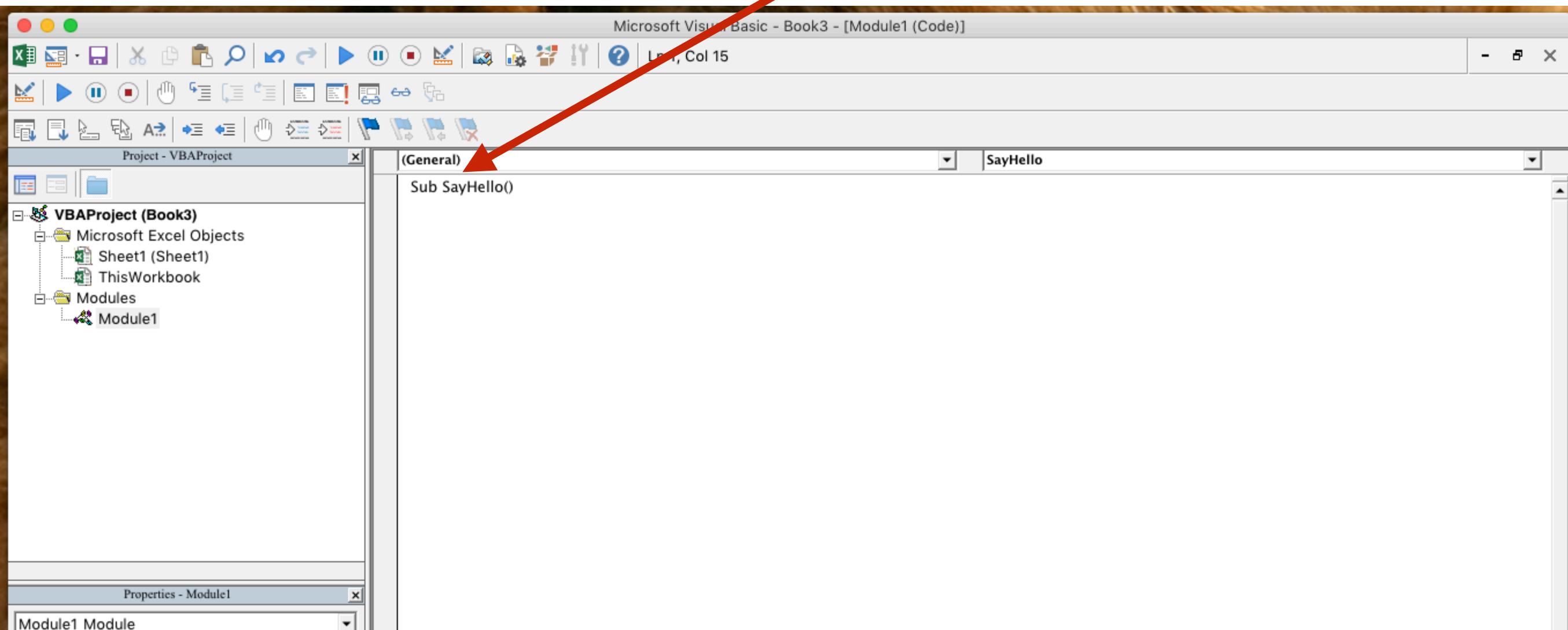


This shows the structure of the code, which is divided into modules

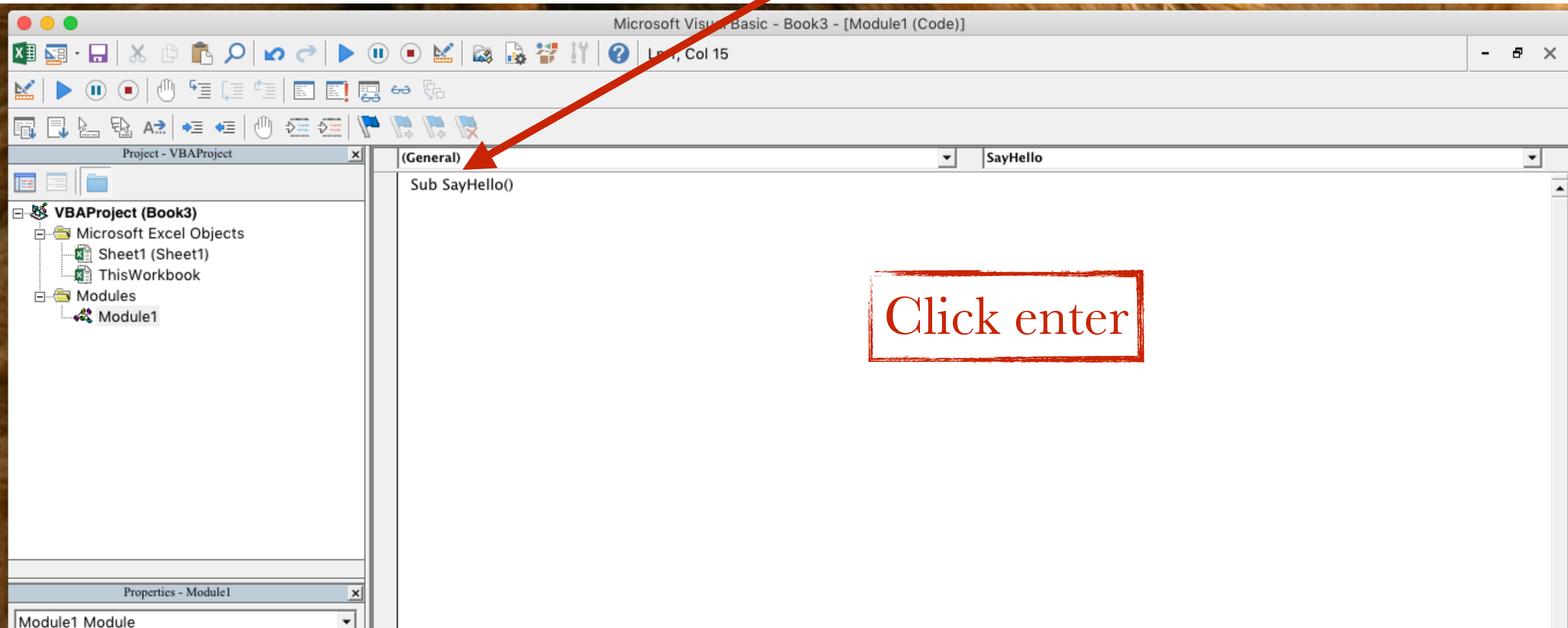




Type:
Sub SayHello()

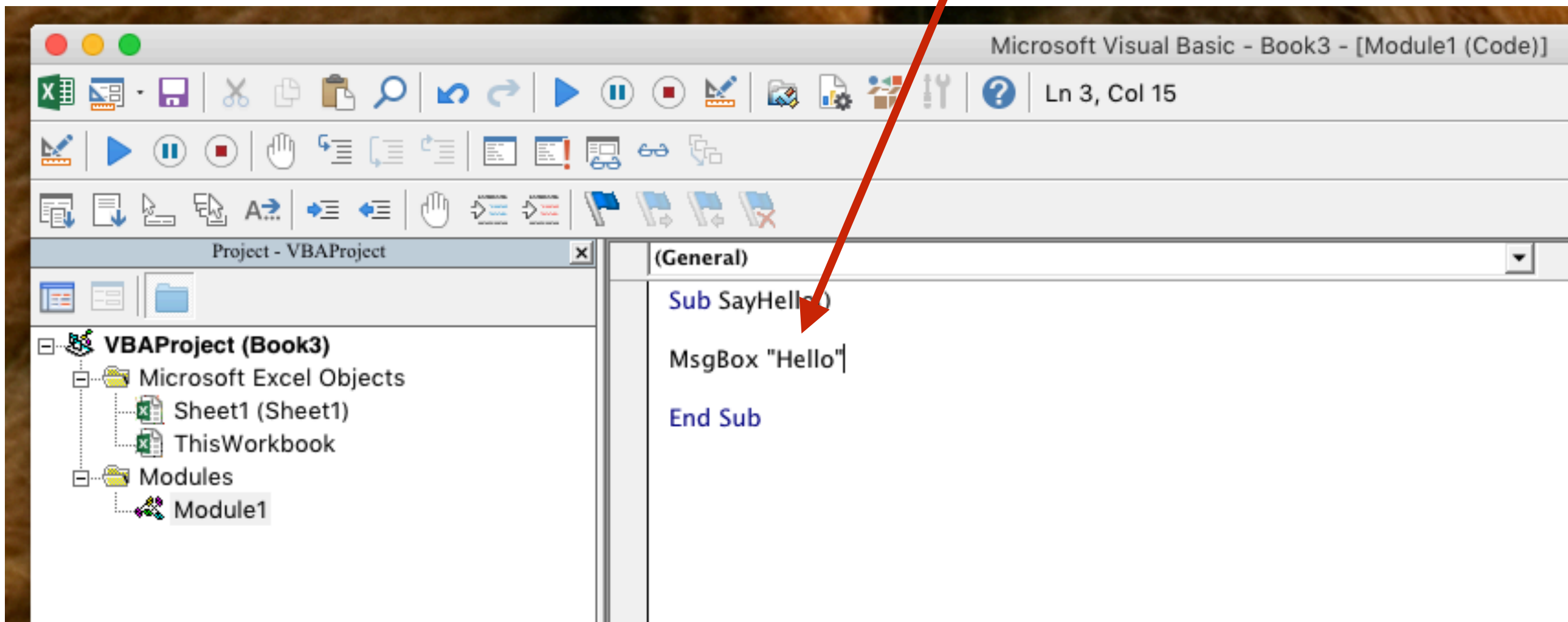


Type:
Sub SayHello()

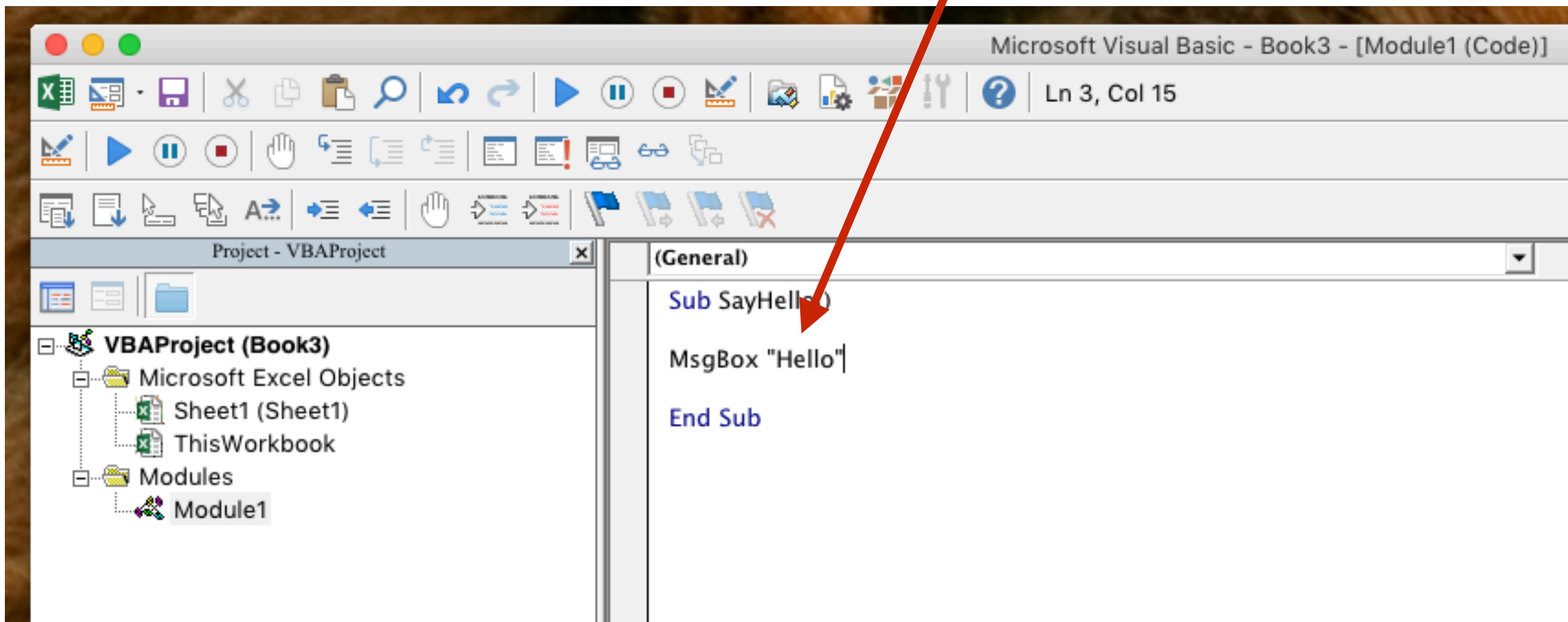


Click enter

This the Message Box command

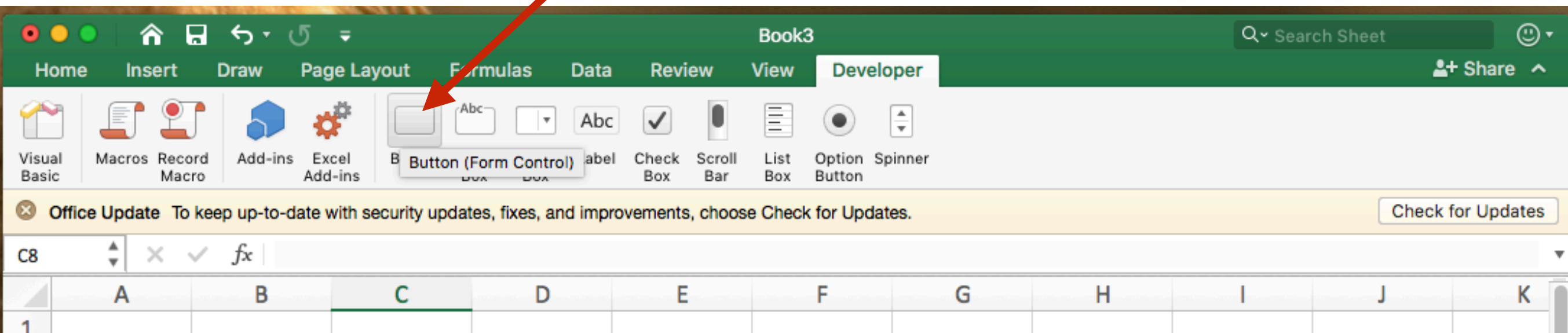


This the Message Box command

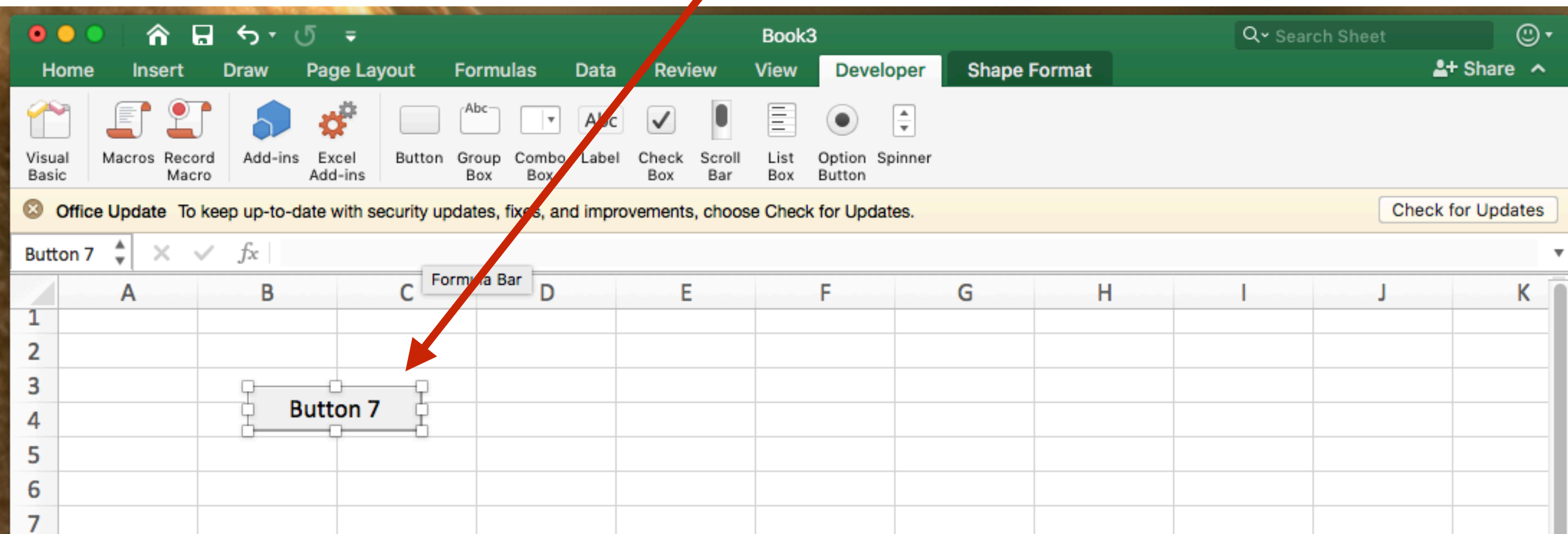


Now we want to assign this macro to a button =>
go back to xls sheet.

Click on Button



Choose a location for the button



Assign the SayHello macro

Assign Macro

Macro name:

SayHello

SayHello

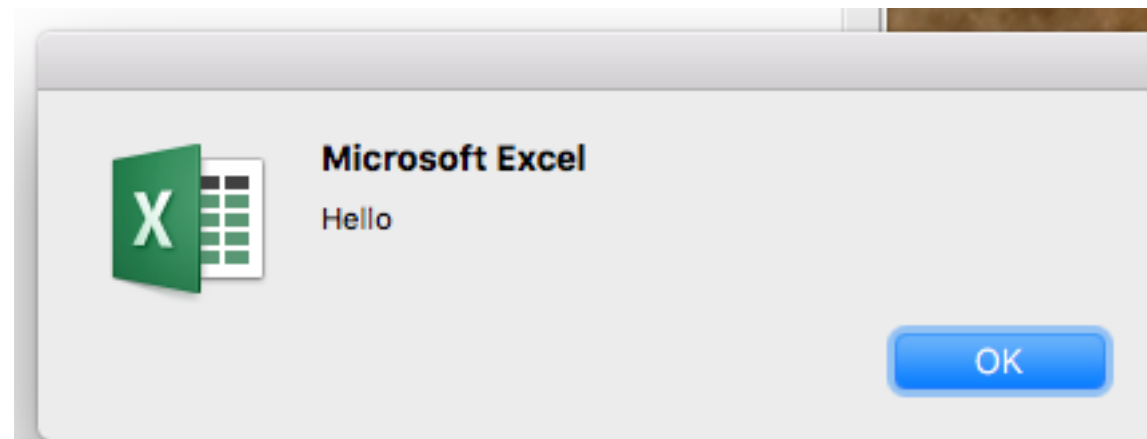
Edit Record...

Macros in: This Workbook

Description:

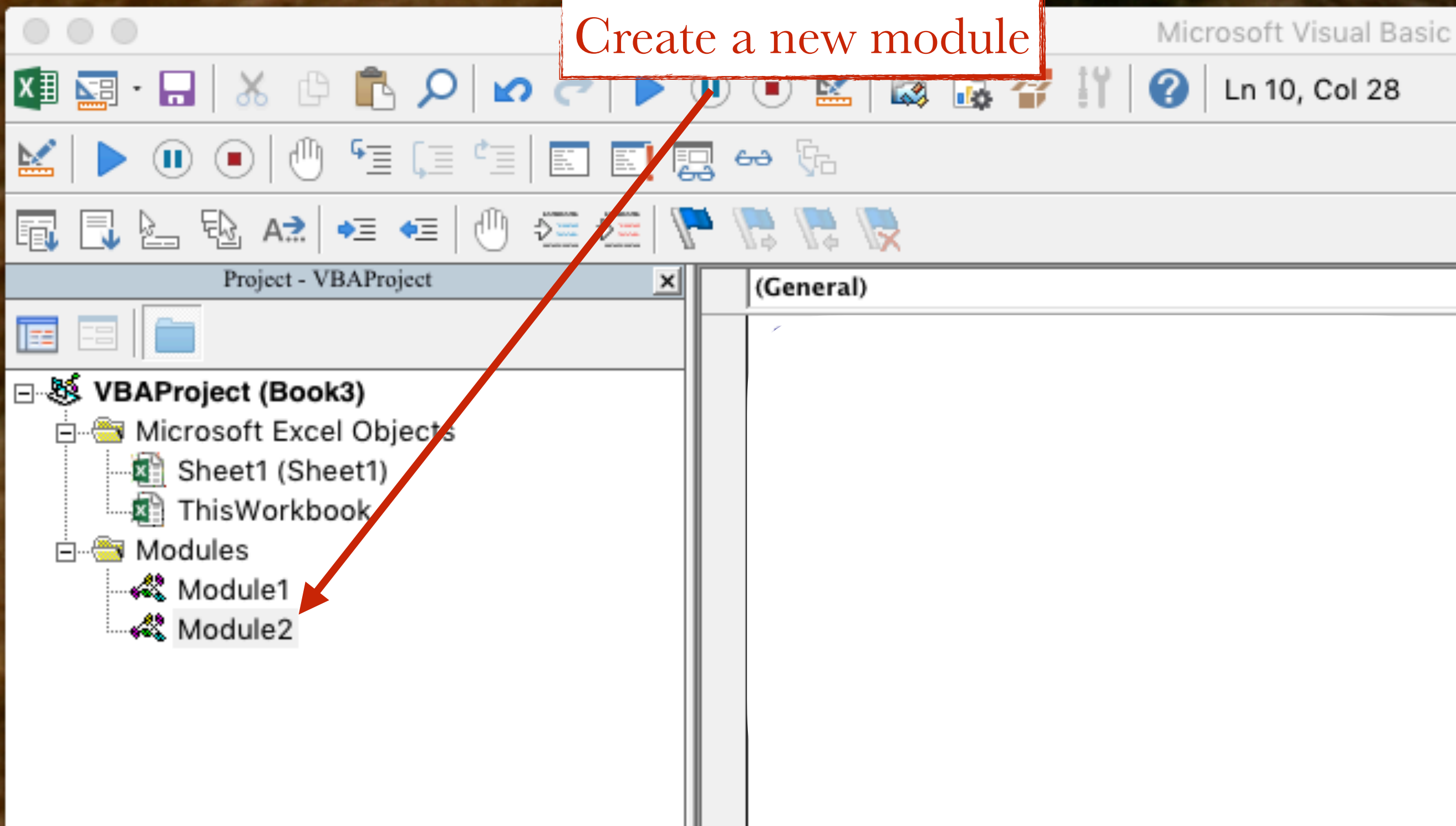
Cancel OK

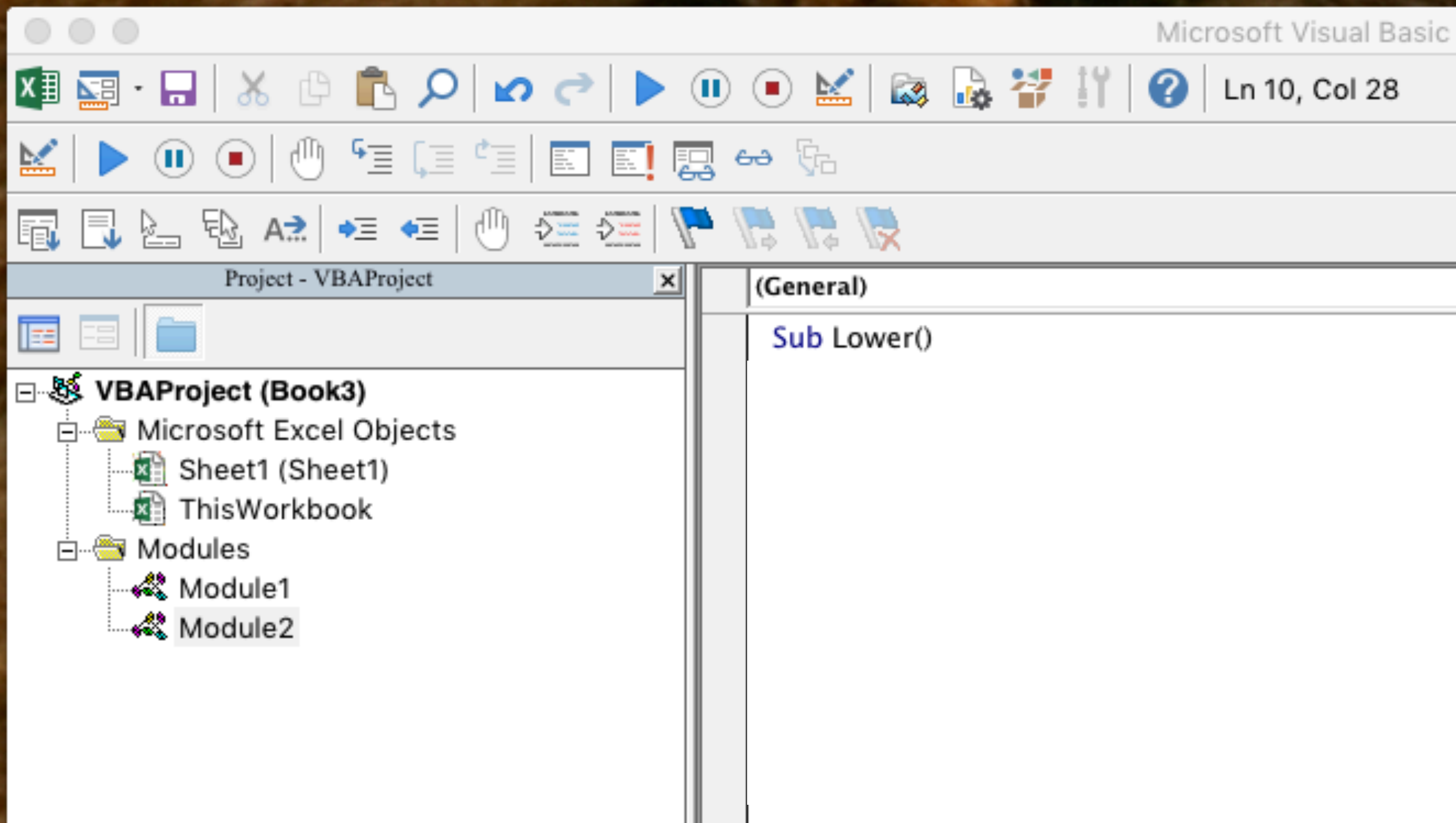
Left click on the Button to see the result

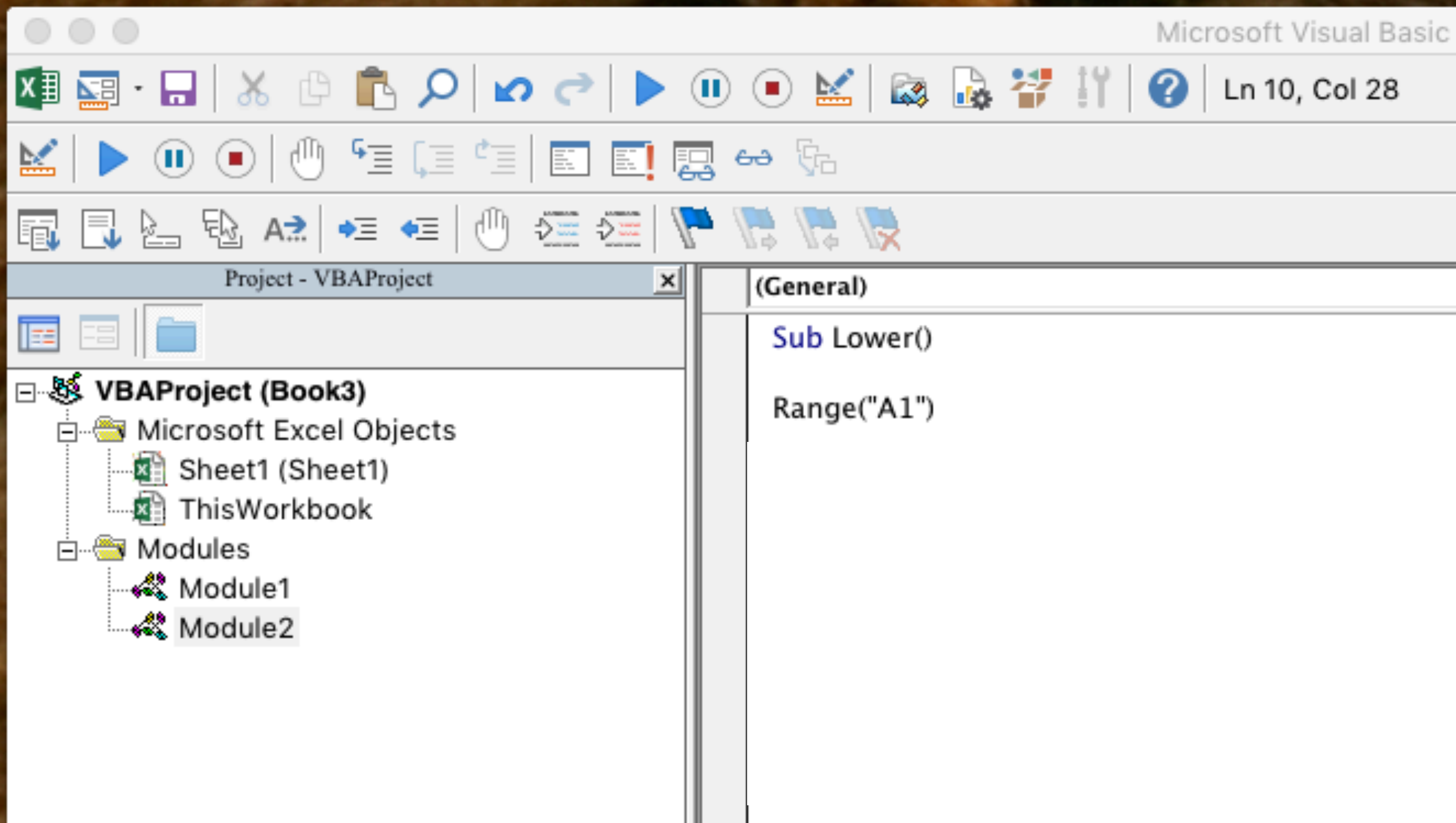


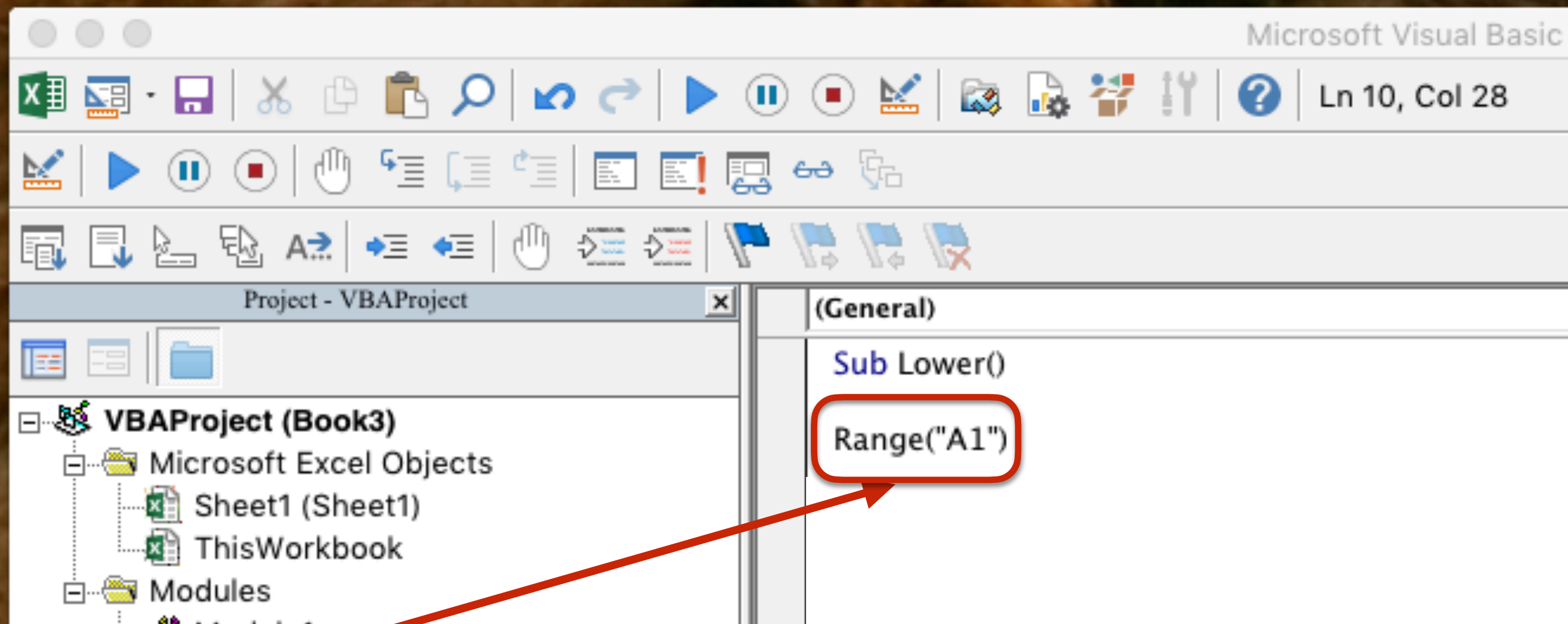
Worksheet Functions

Create a new module

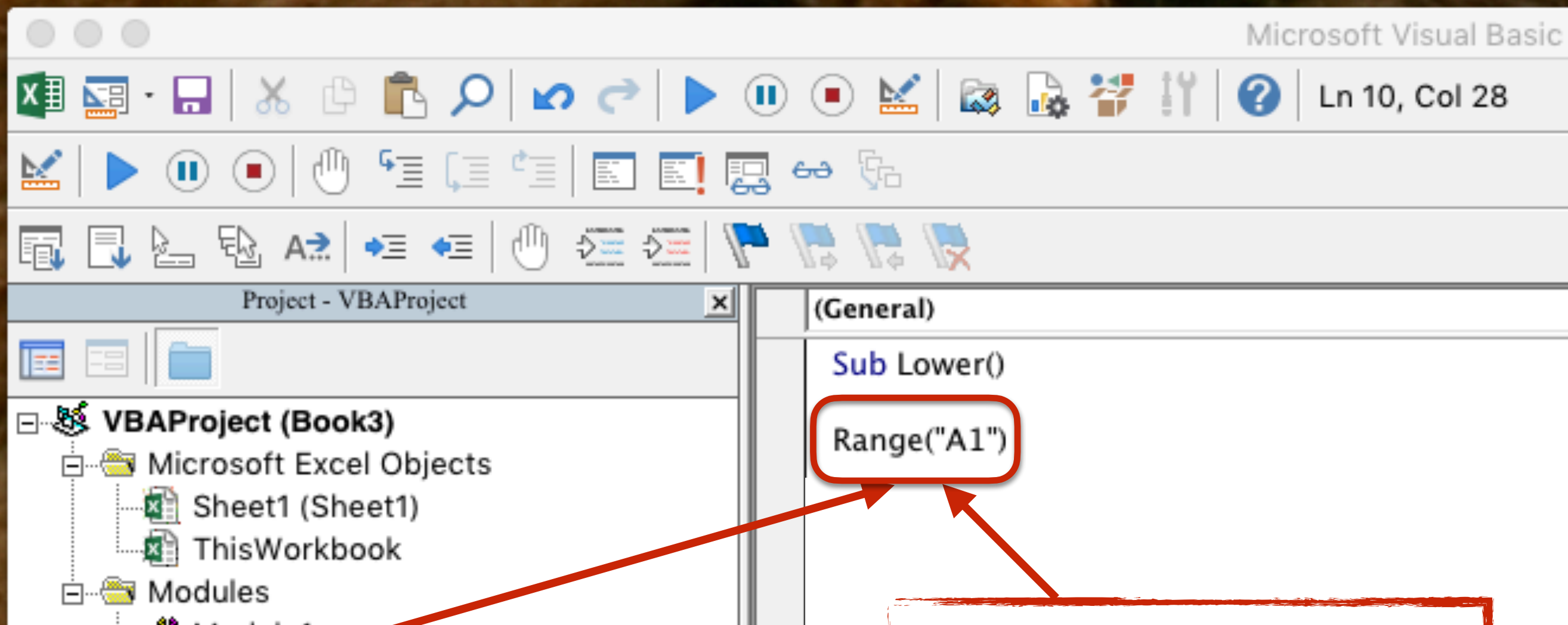






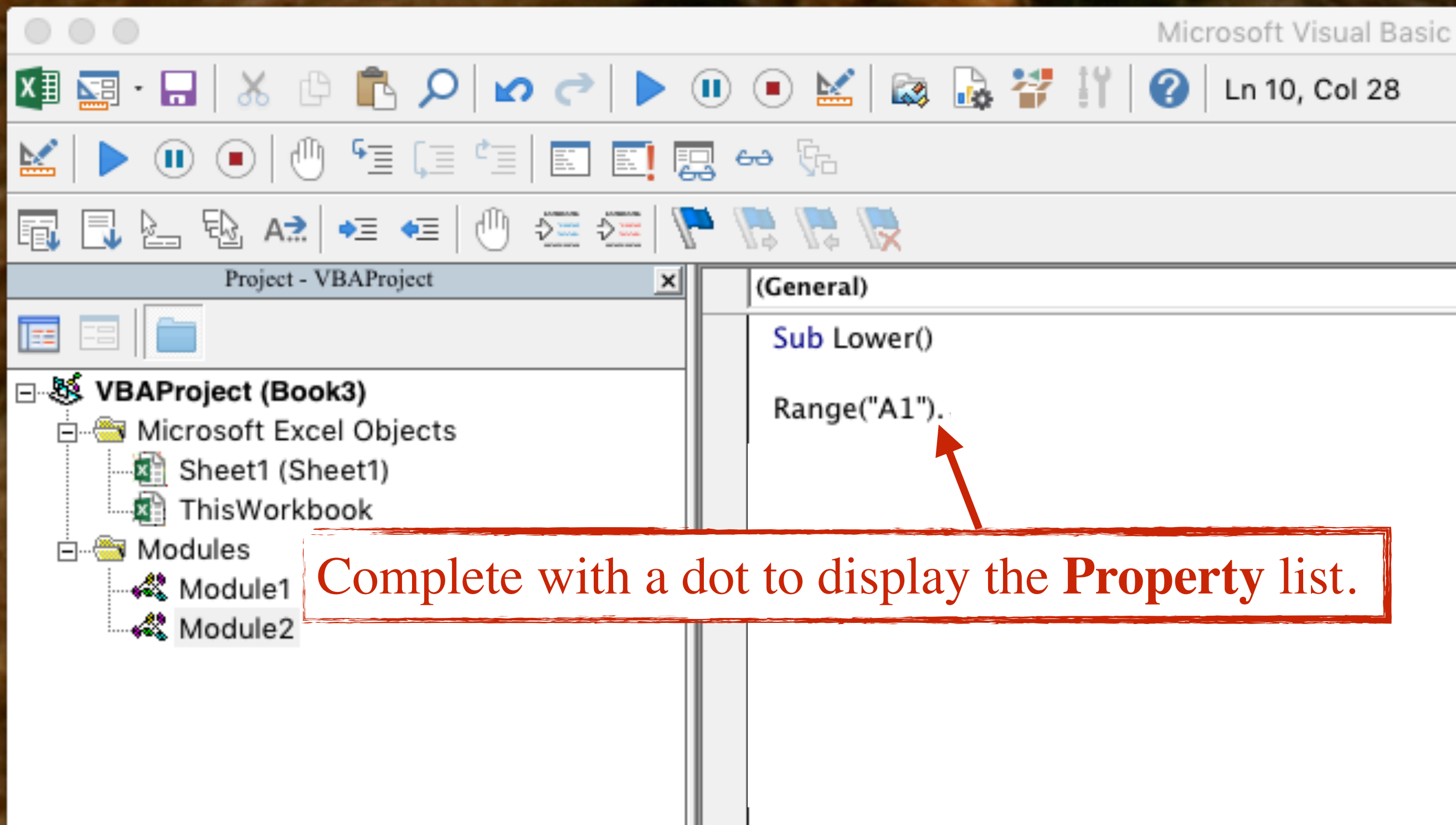


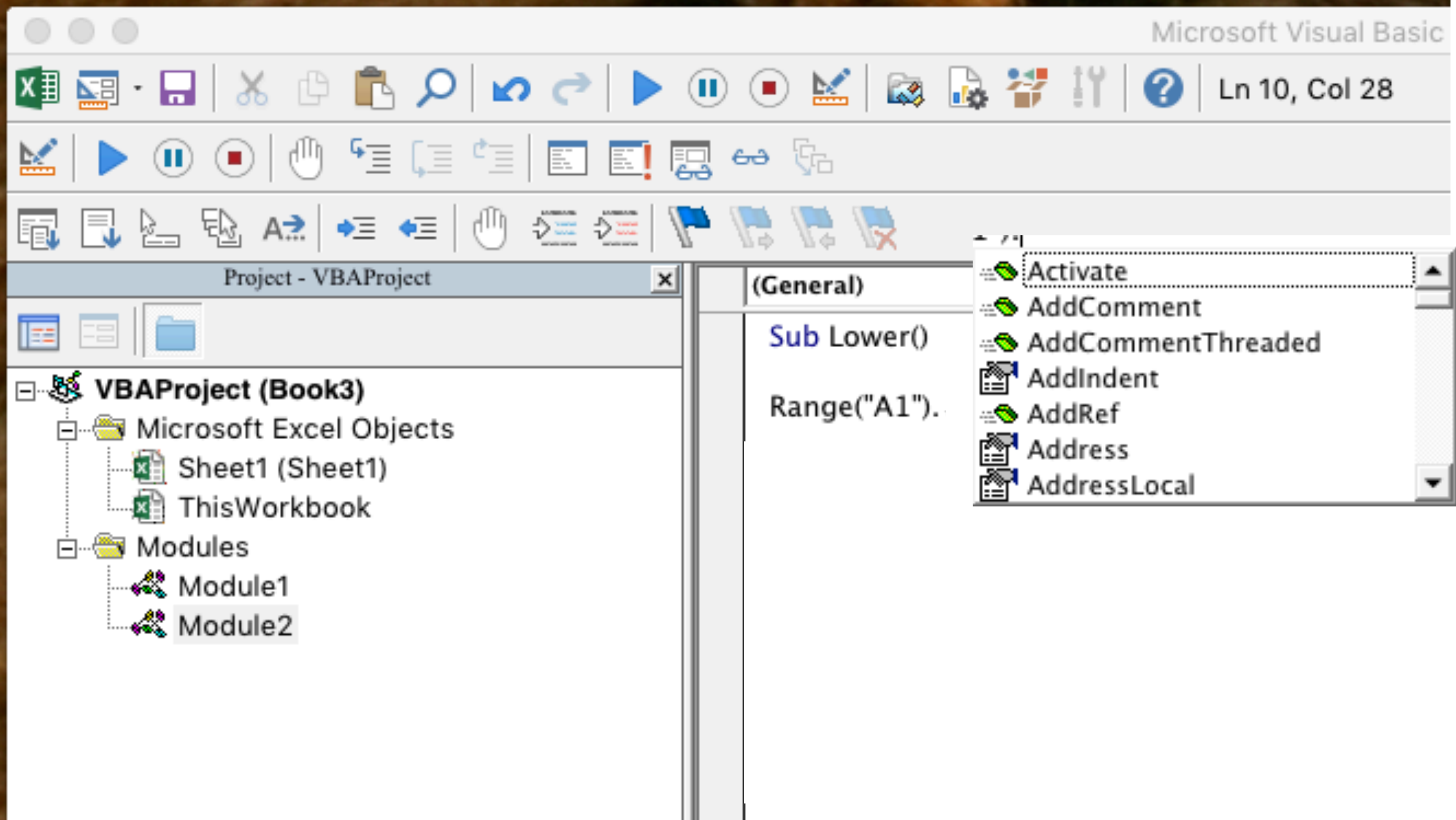
Use the function **Range** (*arg*), where *arg* names the range, to return a **Range** object that represents a single cell or a range of cells.

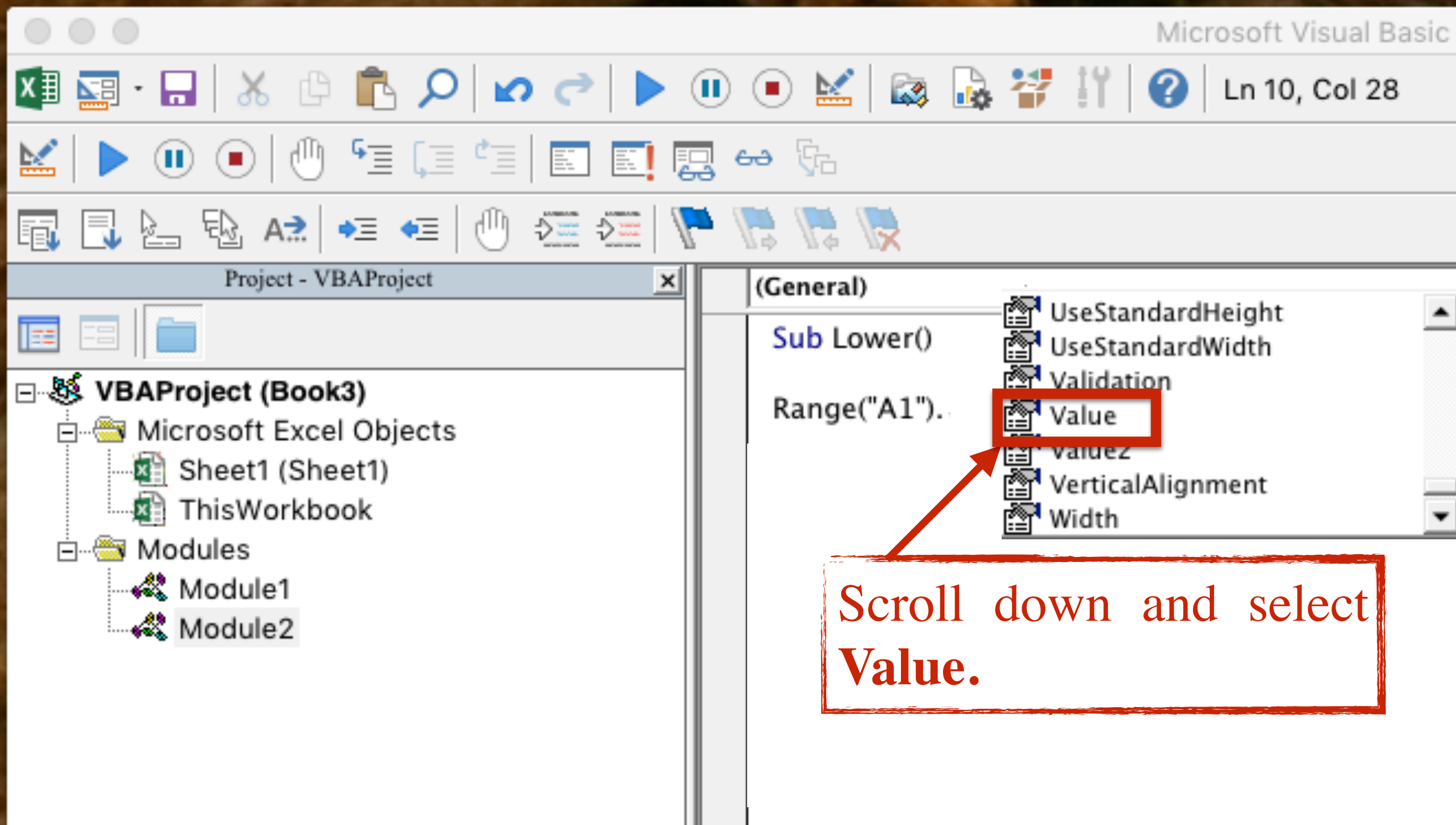


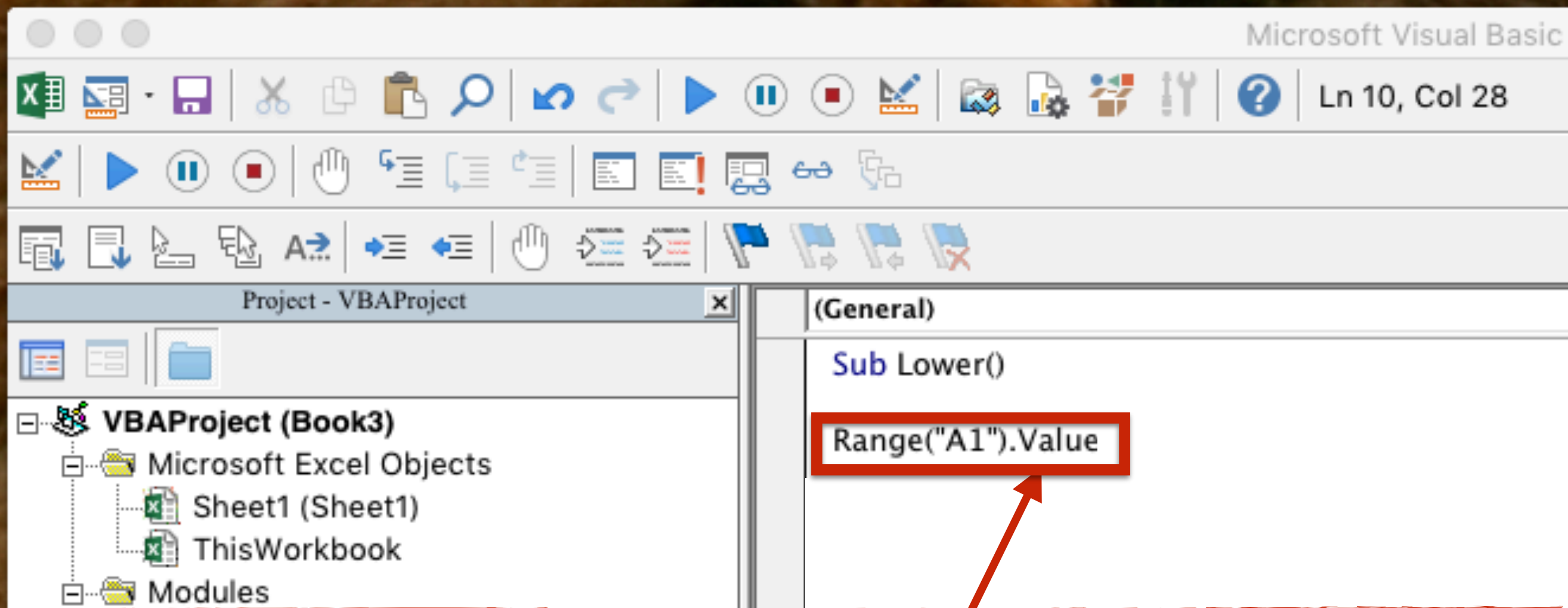
Use the function **Range** (*arg*), where *arg* names the range, to return a **Range** object that represents a single cell or a range of cells.

This **Range** objects now points to the Cell A1.

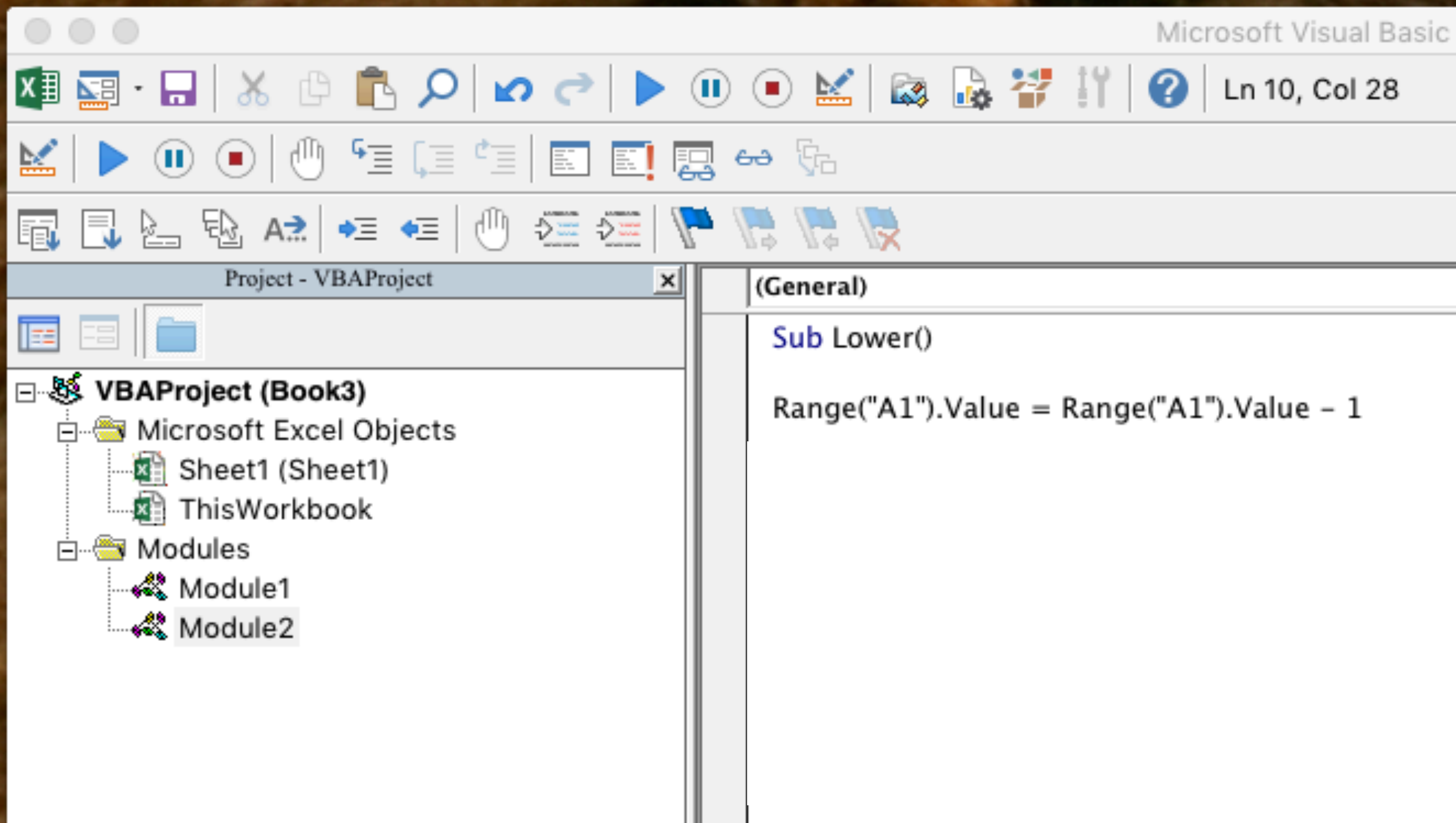




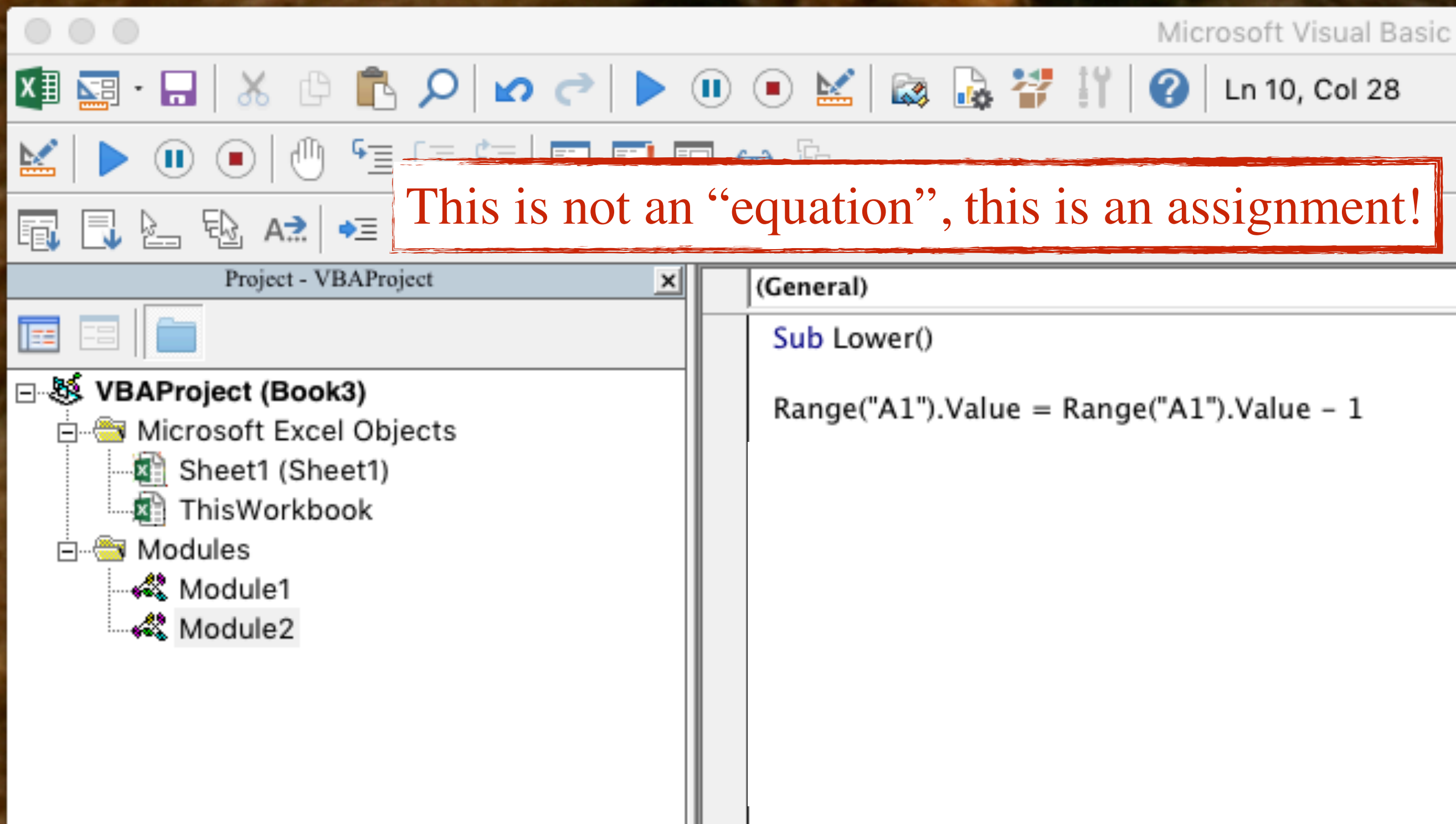


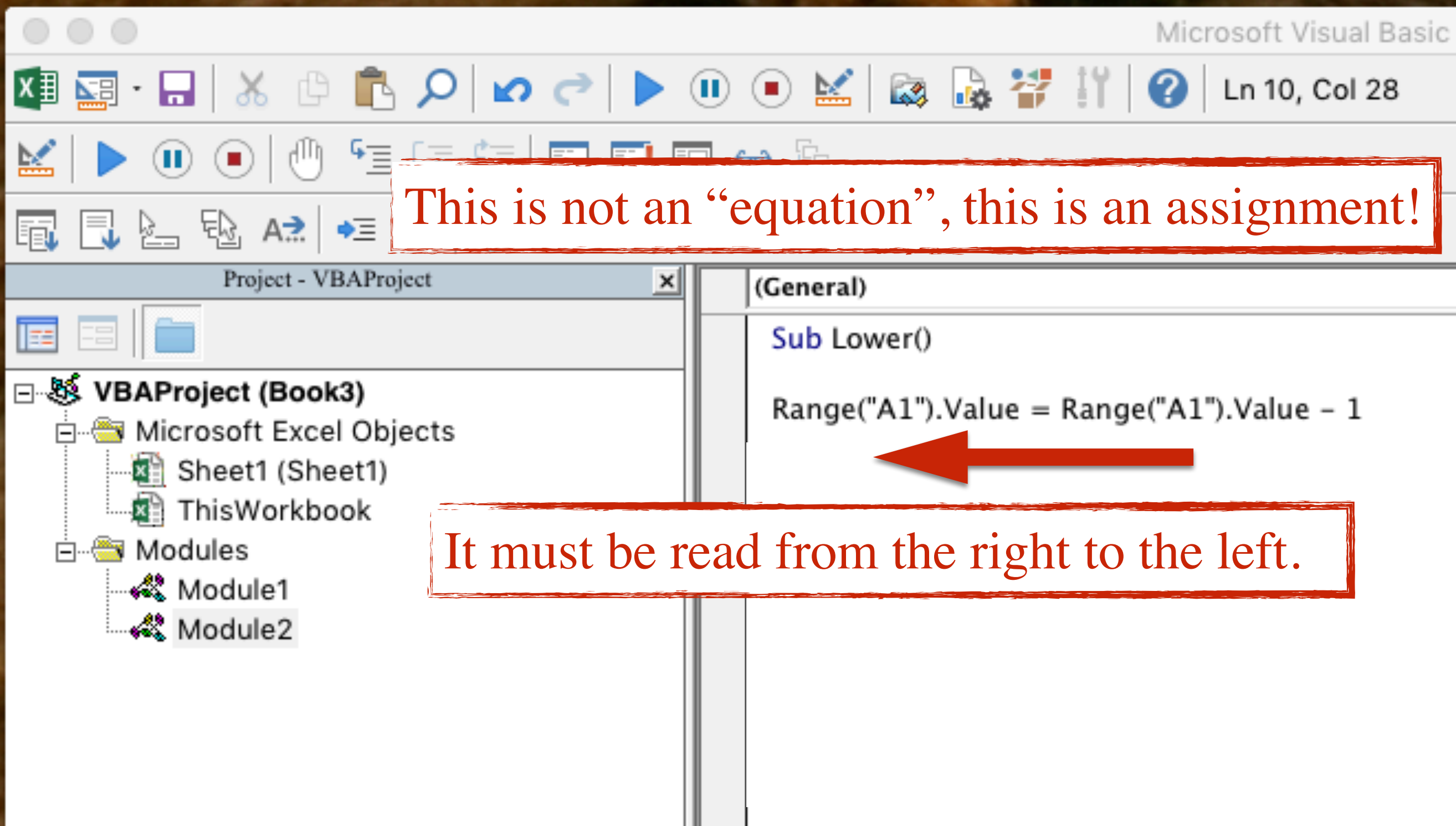


We have now the possibility to modify the value of cell A1, at each execution of the macro.



This is not an “equation”, this is an assignment!





This is not an "equation", this is an assignment!

It must be read from the right to the left.

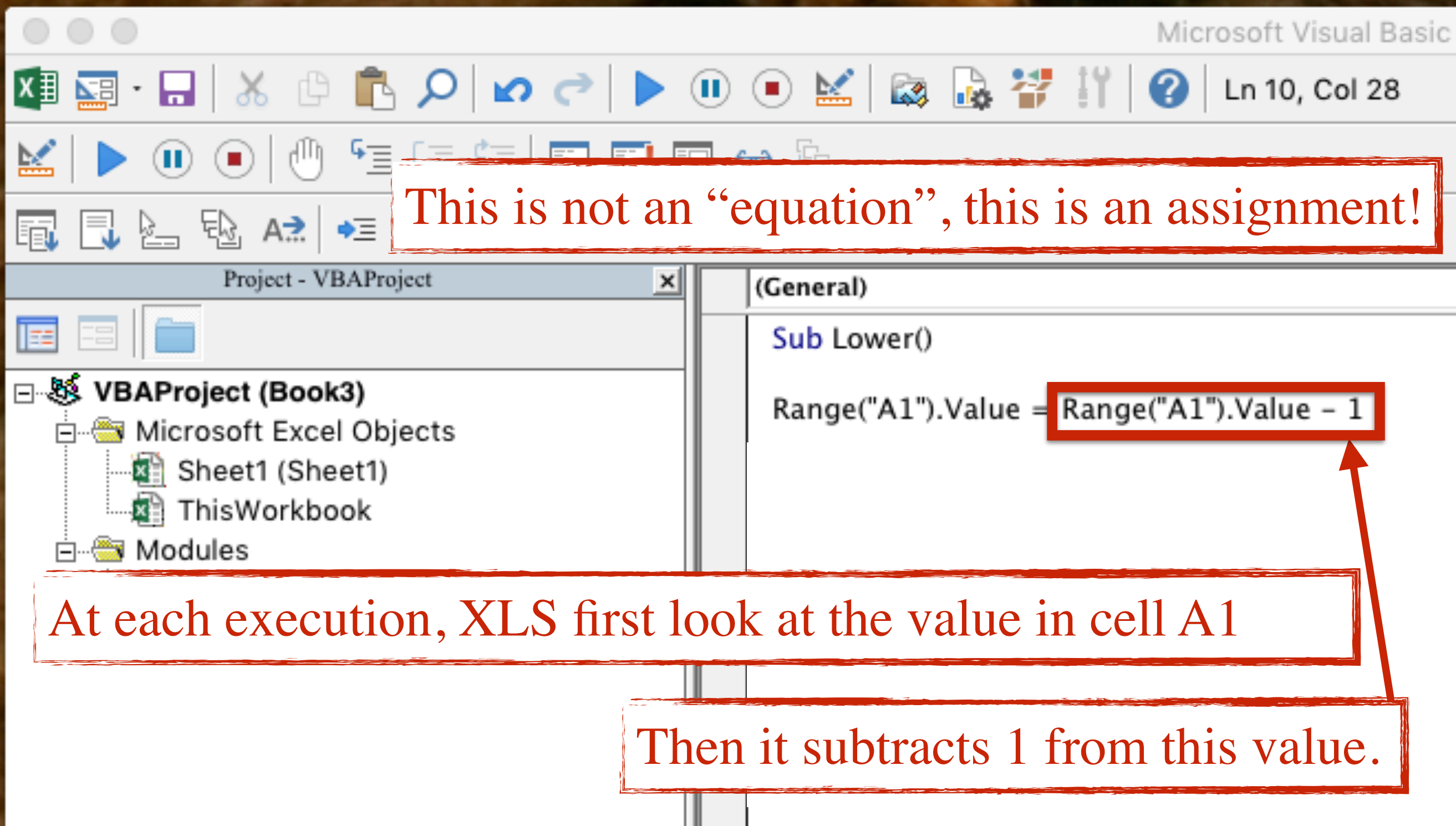
This is not an “equation”, this is an assignment!

Sub Lower()

Range("A1").Value = Range("A1").Value - 1

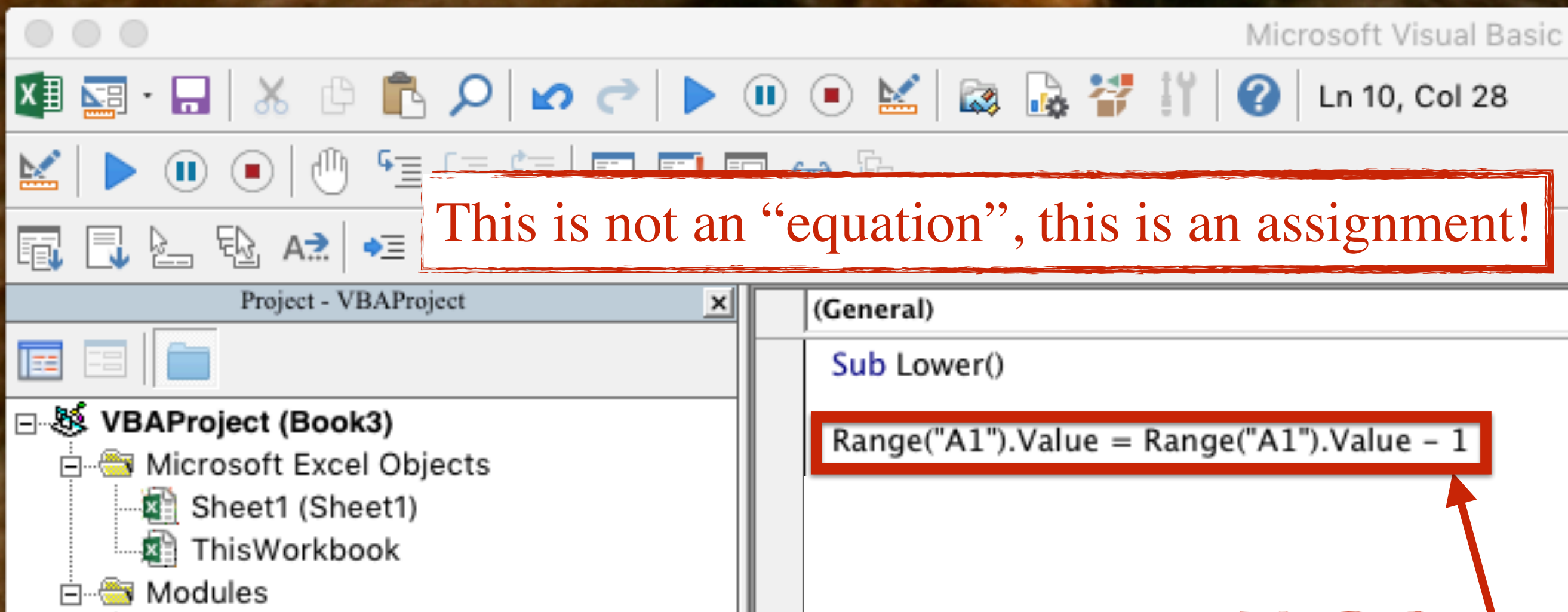
At each execution, XLS first look at the value in cell A1

This is not an “equation”, this is an assignment!



At each execution, XLS first look at the value in cell A1

Then it subtracts 1 from this value.

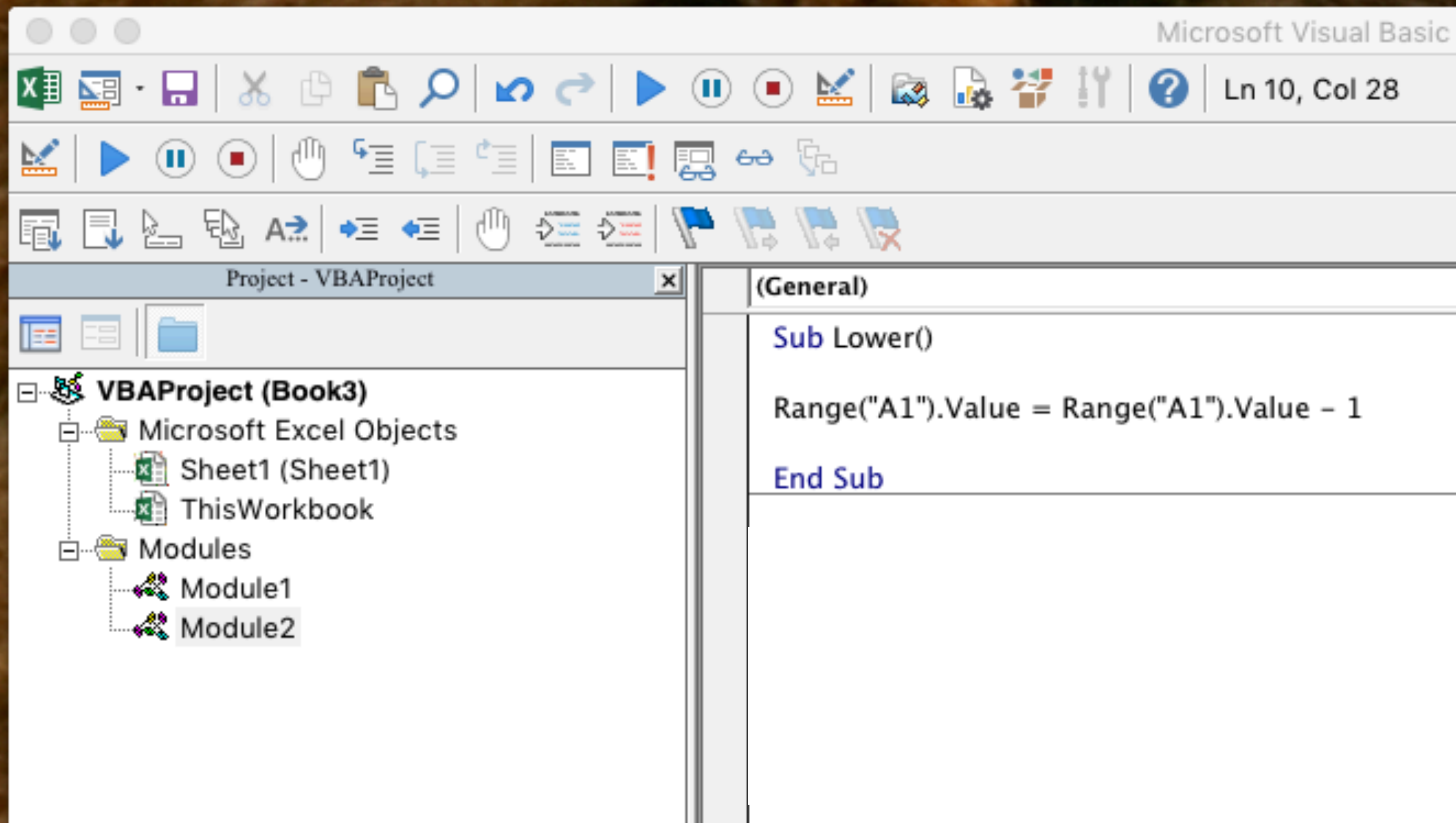


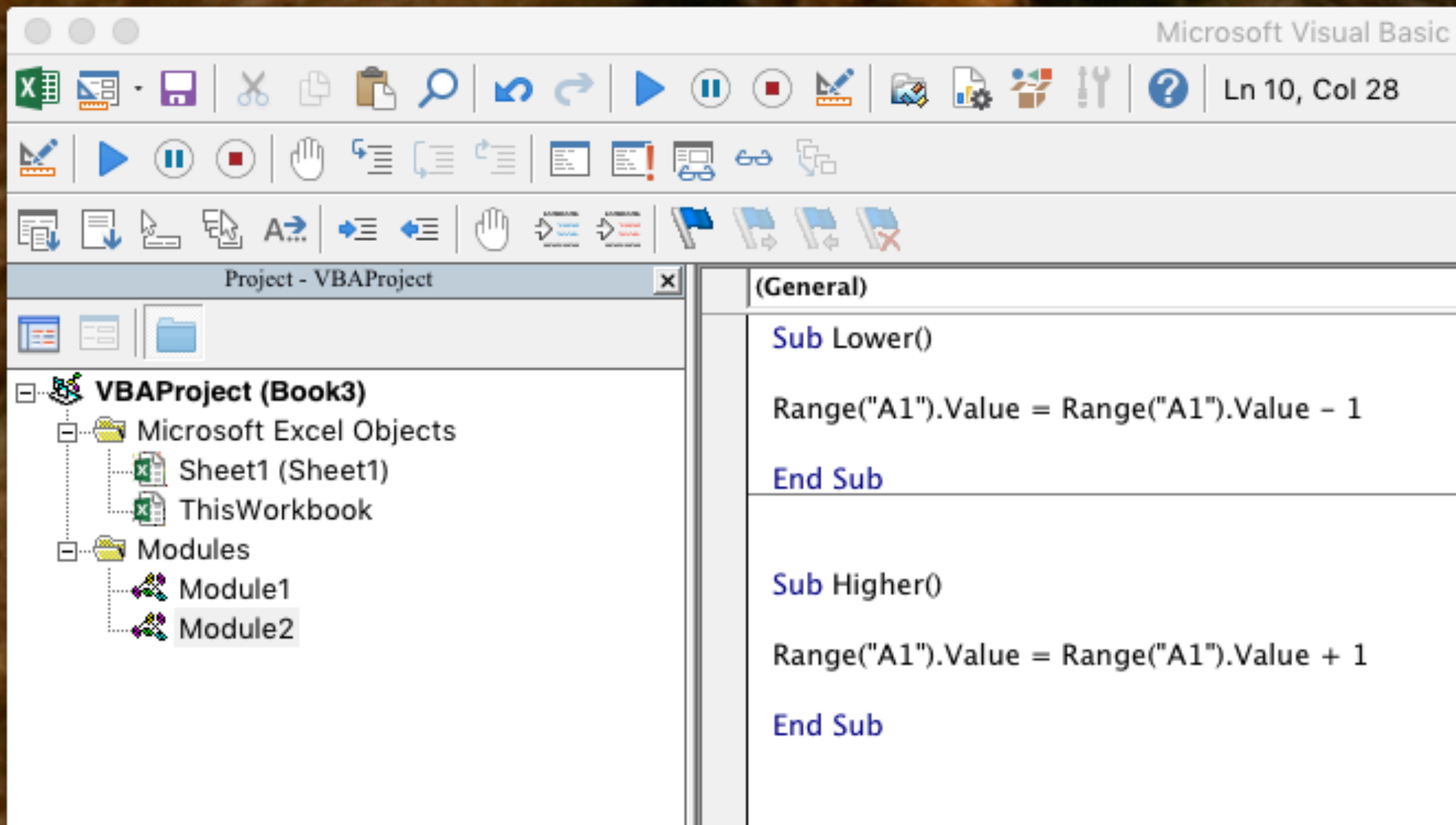
This is not an “equation”, this is an assignment!

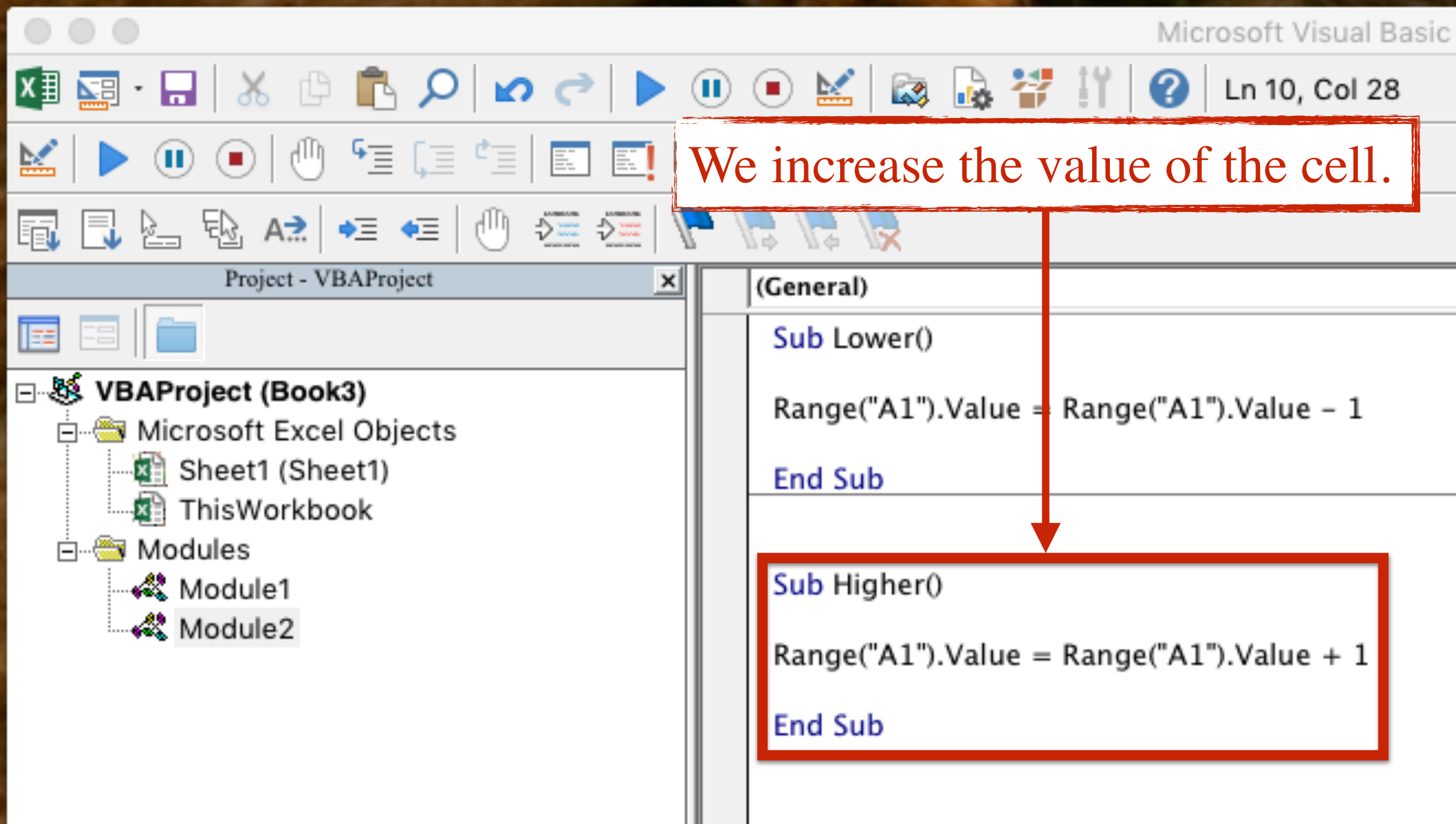
At each execution, XLS first look at the value in cell A1

Then it subtracts 1 from this value.

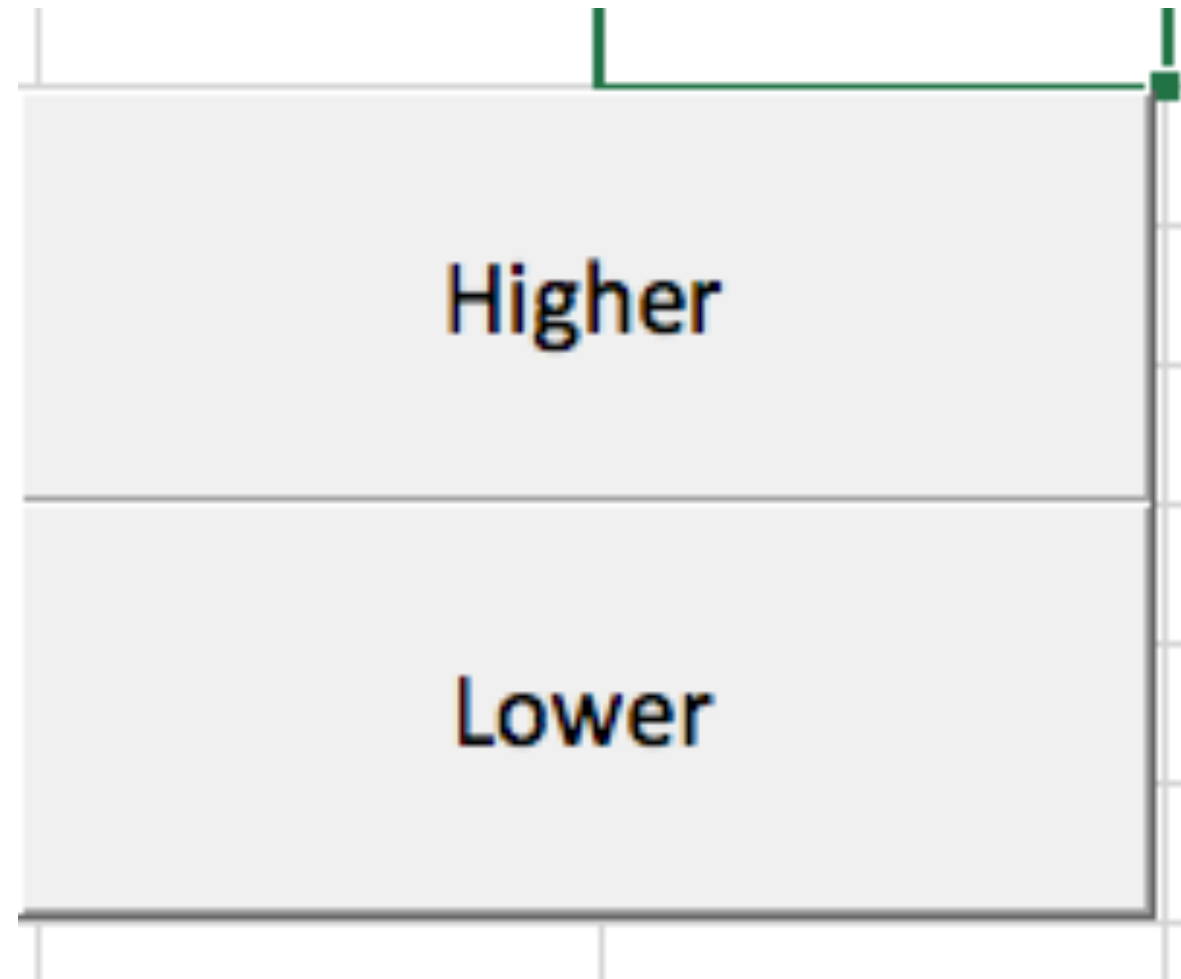
Finally, it assigns the value computed to the cell A1







Create two new buttons and attach them to the corresponding the macros



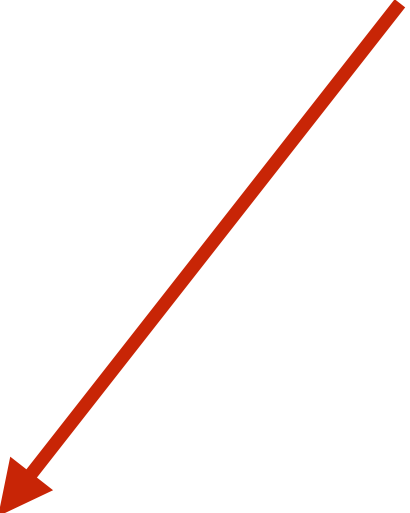
Play with the buttons!

```
Sub multiple()
```

```
Range("A1:A10").Value = Range("B1").Value + 1
```

```
End Sub
```

A range object that represents all the cell from A1 to A10



Sub multiple()

Range("A1:A10").Value = Range("B1").Value + 1

End Sub

A range object that represents all the cell from A1 to A10

```
graph TD; A["A range object that represents all the cell from A1 to A10"] --> B["Range(\"A1:A10\").Value"]; C["We assign to all of them the value in cell B1 +1"] --> D["Range(\"B1\").Value + 1"]
```

Sub multiple()

Range("A1:A10").Value = Range("B1").Value + 1

End Sub

We assign to all of them the value in cell B1 +1

Book2

Home Insert Draw Page Layout Formulas Data Review View Developer

Visual Basic Macros Use Relative References Add-ins Excel Add-ins Button Group Combo Label Checkbox Scroll List Option Spinner

B1 x ✓ fx 6

	A	B	C	D
1	7	6		
2	7			
3	7			
4	7	Button 1		
5	7			
6	7			
7	7			
8	7			
9	7			
10	7			

Whatever we put in B1, after each execution the cell from A1 to A10 will contain the value in B1 plus one.

Sheet1

Ready 200%

A range object that represents all the cell from A1 to B10

(General)

Sub multiple()

Range("A1:B10").Value = Range("B1").Value + 1

End Sub

We assign to all of them the value in cell B1 +1

Book2

Search Sheet

Home Insert Draw Page Layout Formulas Data Review View Developer

Visual Basic Macros Use Relative References Add-ins Excel Add-ins Button Group Box Combo Box Label Checkbox Scroll Bar List Box Option Button Spinner

B4

	A	B	C	D
1		1		
2				
3				
4				
5				
6	Associate the button with the marco and put a 1 in cell B1			Button 2
7				
8				
9				
10				

Sheet1

Ready

200%

Book2

Search Sheet

Home Insert Draw Page Layout Formulas Data Review View Developer

Visual Basic Macros Use Relative References Add-ins Excel Add-ins Button Group Box Combo Box Label Checkbox Scroll Bar List Box Option Button Spinner

B4

	A	B	C	D
1		1		
2				
3				
4				
5				
6	Associate the button with the marco and put a 1 in cell B1			Button 2
7				
8	Click the button to execute the routine			
9				
10				

Sheet1

Ready

200%

	A	B	C	D
1	2	2		
2	2	2		
3	2	2		
4	2	2		
5	2	2		
6	2	2	Button 2	
7	2	2		
8	2	2		
9	2	2		
10	2	2		

Microsoft Visual Basic - Bo

Ln 3, Col 38

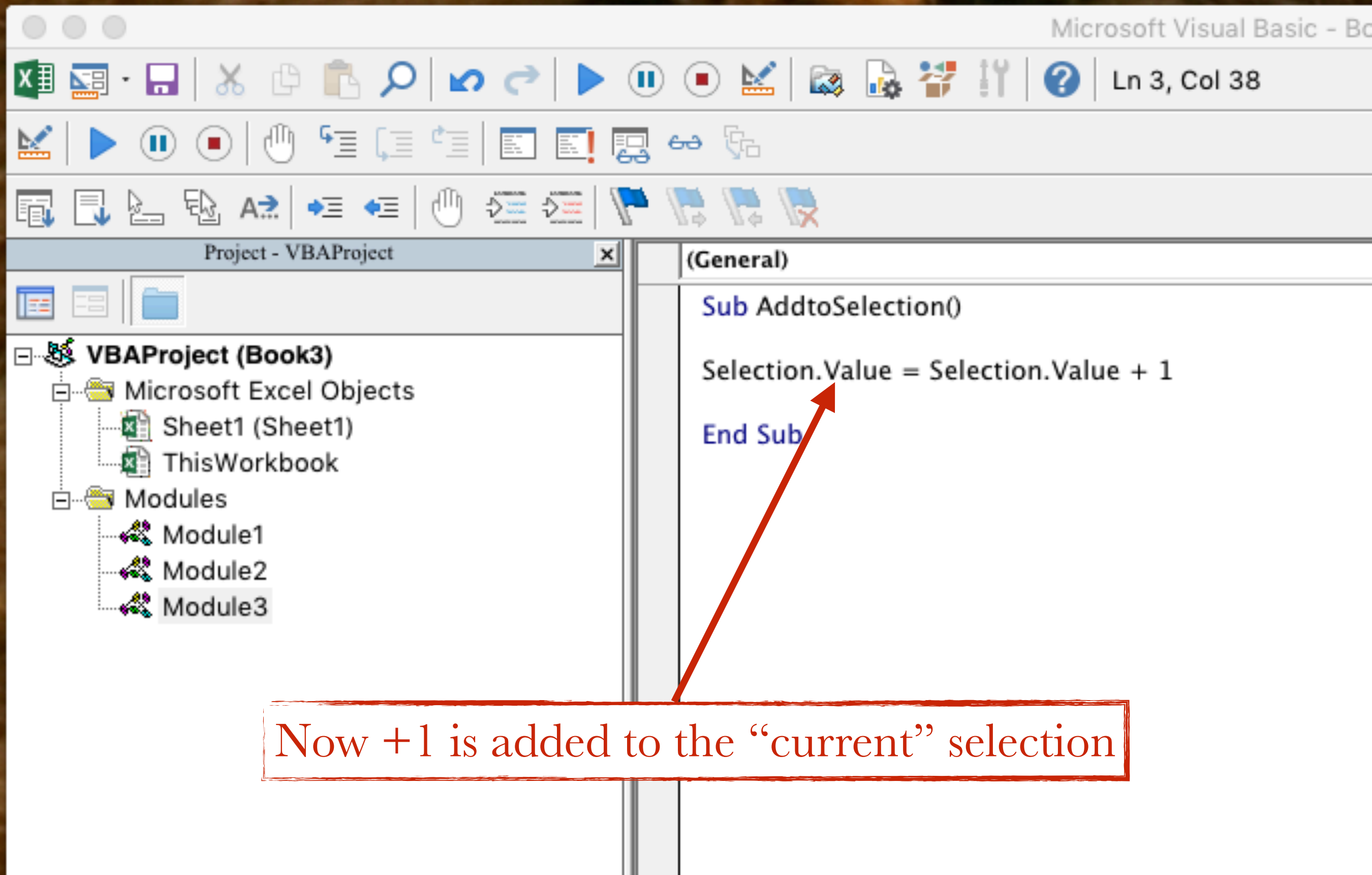
Project - VBAProject

VBAProject (Book3)

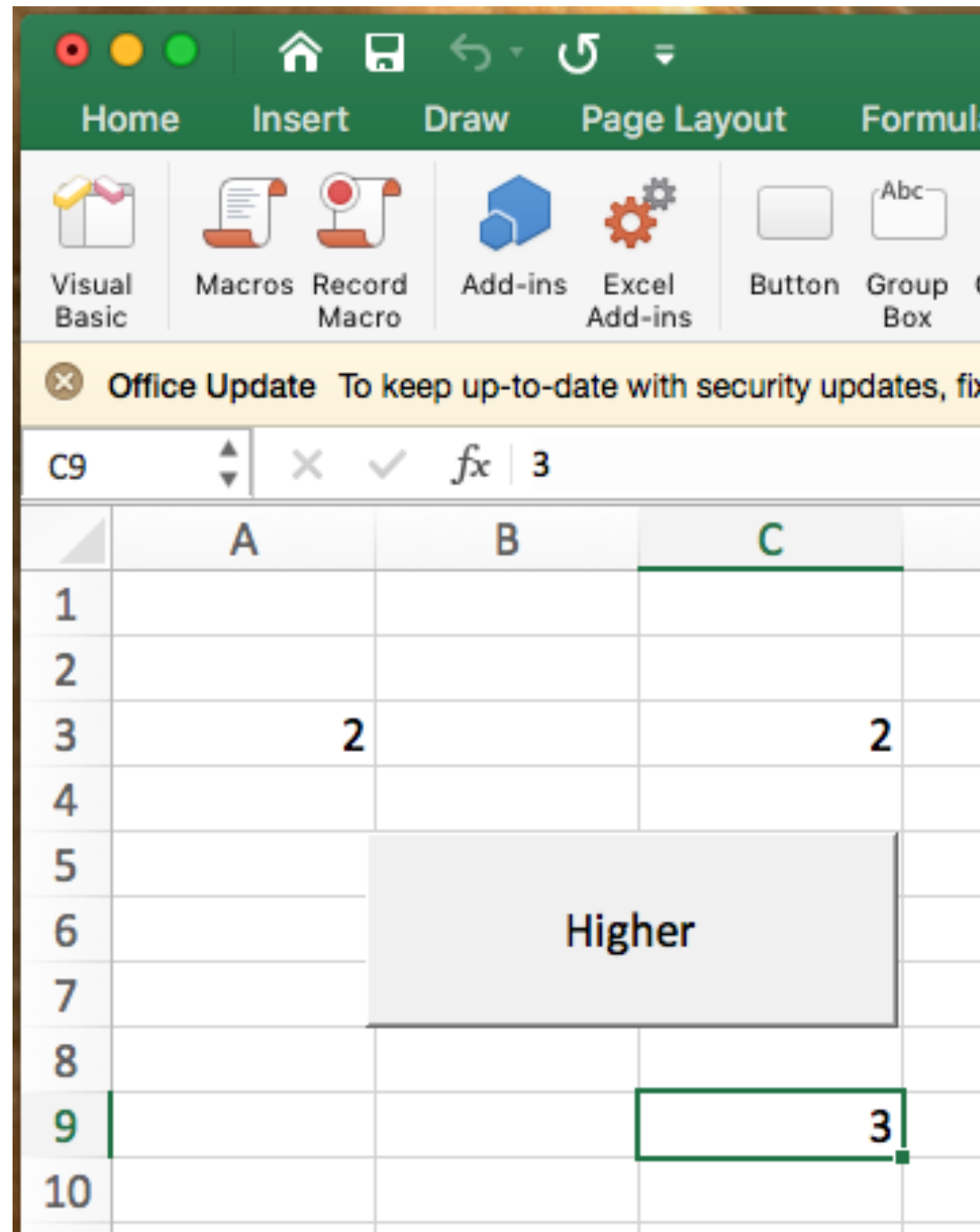
- Microsoft Excel Objects
 - Sheet1 (Sheet1)
 - ThisWorkbook
- Modules
 - Module1
 - Module2
 - Module3

(General)

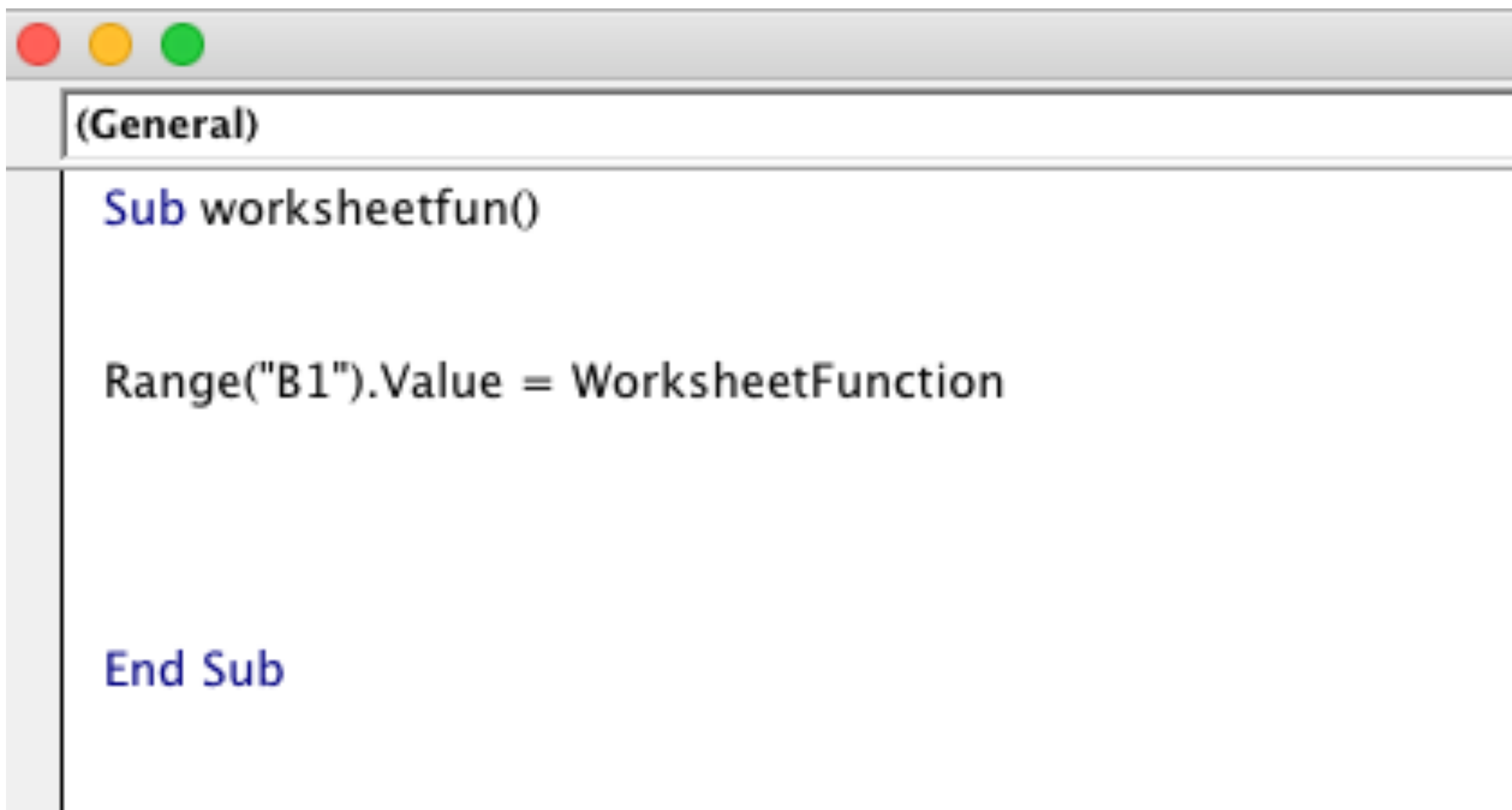
```
Sub AddtoSelection()  
  
Selection.Value = Selection.Value + 1  
  
End Sub
```



Create a new button and attach it to the new macro



Play with the button



(General)

Sub worksheetfun()

Range("B1").Value = WorksheetFunction.

End Sub

- Count
- CountA
- CountBlank
- CountIf
- CountIfs
- CoupDayBs
- CoupDays

(General)

```
Sub worksheetfun()
```

```
Range("B1").Value = WorksheetFunction.
```

```
End Sub
```

- Count
- CountA
- CountBlank
- CountIf
- CountIfs
- CoupDayBs
- CoupDays



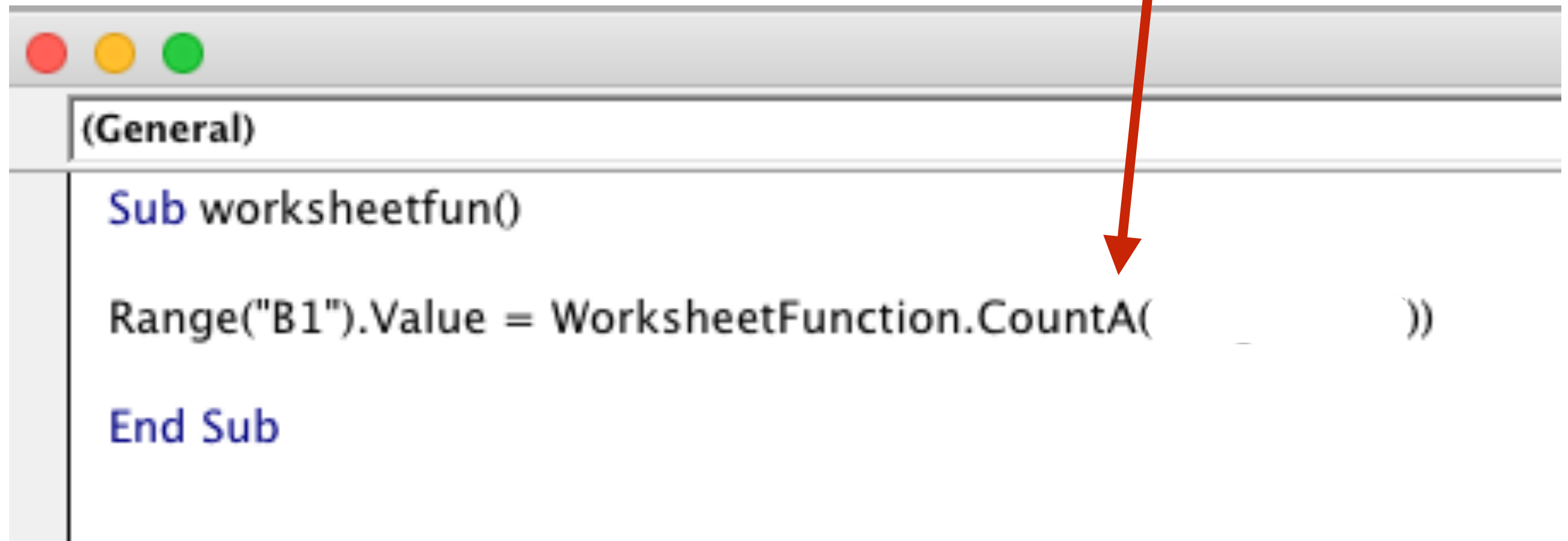
(General)

```
Sub worksheetfun()
```

```
Range("B1").Value = WorksheetFunction.CountA(            ))
```

```
End Sub
```

The **COUNTA** function counts the number of cells that are not empty in a range.

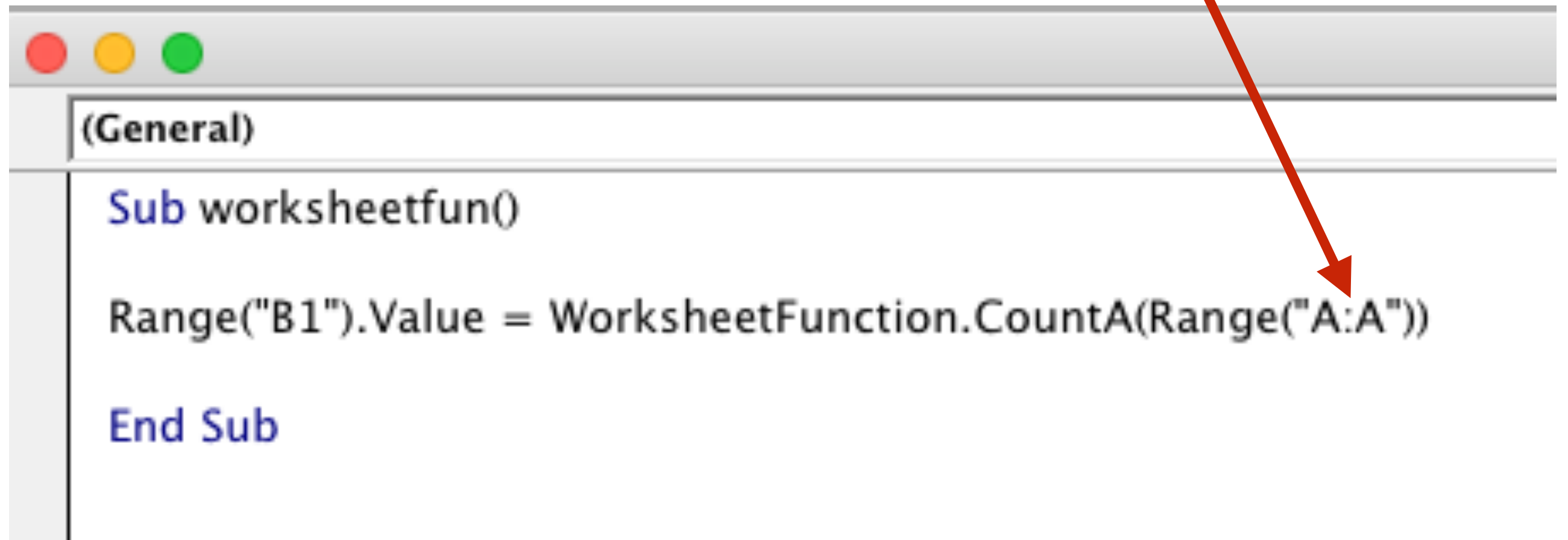


The image shows a VBA editor window with a standard macOS-style title bar (red, yellow, and green buttons). The window has a tab labeled "(General)". The code editor contains the following VBA code:

```
Sub worksheetfun()  
Range("B1").Value = WorksheetFunction.CountA(  
  
))  
End Sub
```

A red arrow points from the text box above to the `CountA` function in the code.

We pass to the CountA function the object Range("A:A"), i.e. the content of the first column



```
(General)  
  
Sub worksheetfun()  
  
    Range("B1").Value = WorksheetFunction.CountA(Range("A:A"))  
  
End Sub
```


Example:

A	B
a	11
12	
3e	
44	
4	
4	
4	
e	
s	
Doppio	
234	

Variables

Variable could, in principle, not be declared.

Variable could, in principle, not be declared.

If you use a variable that you haven't declare before, VBA will use a standard format for it.

Variable could, in principle, not be declared.

If you use a variable that you haven't declare before, VBA will use a standard format for it.

This could be ok for very simple and short codes.

Variable could, in principle, not be declared.

If you use a variable that you haven't declare before, VBA will use a standard format for it.

This could be ok for very simple and short codes.

But it could be a serious bug for complex codes!



Variable could, in principle, not be declared.

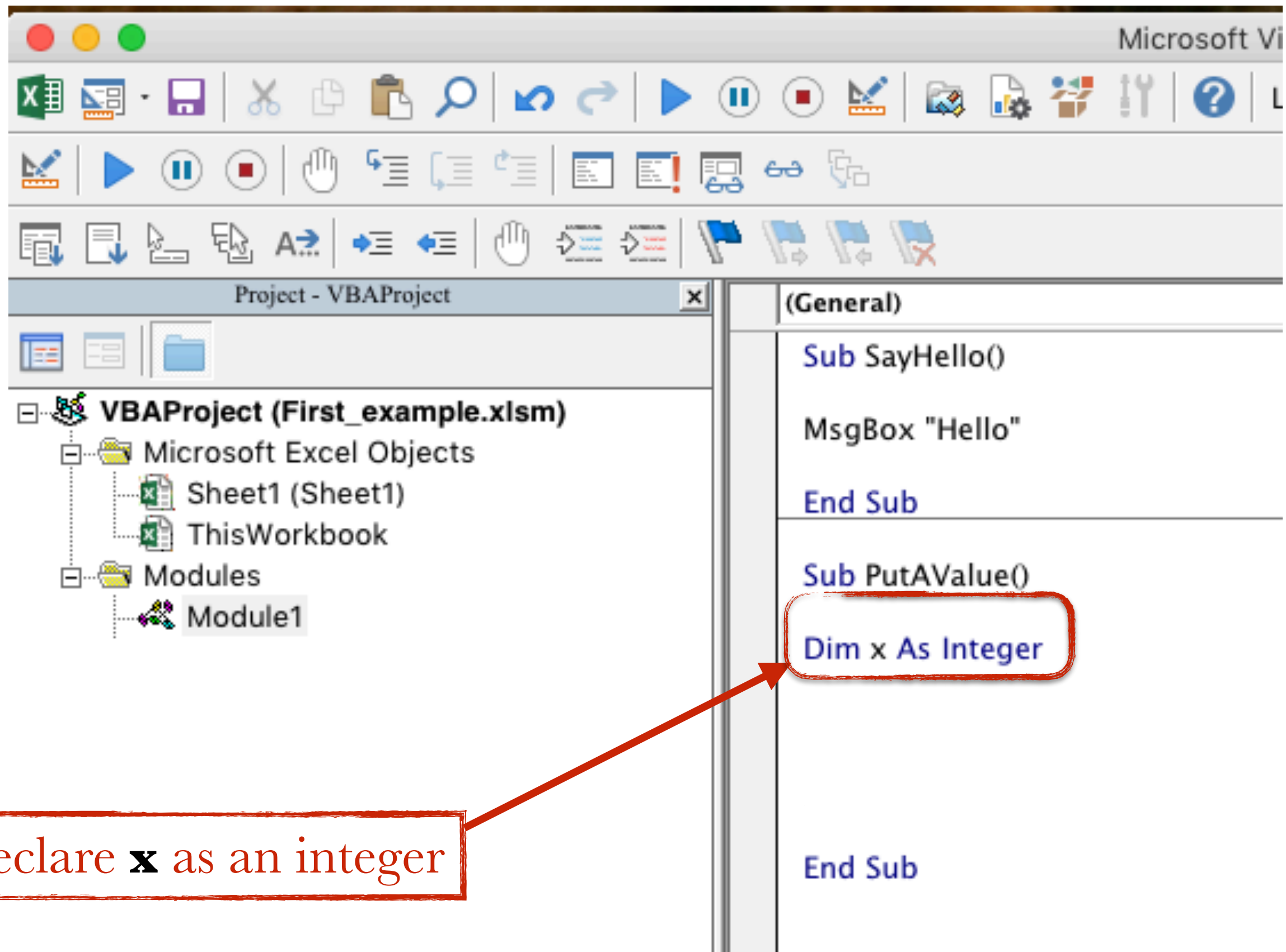
If you use a variable that you haven't declare before, VBA will use a standard format for it.

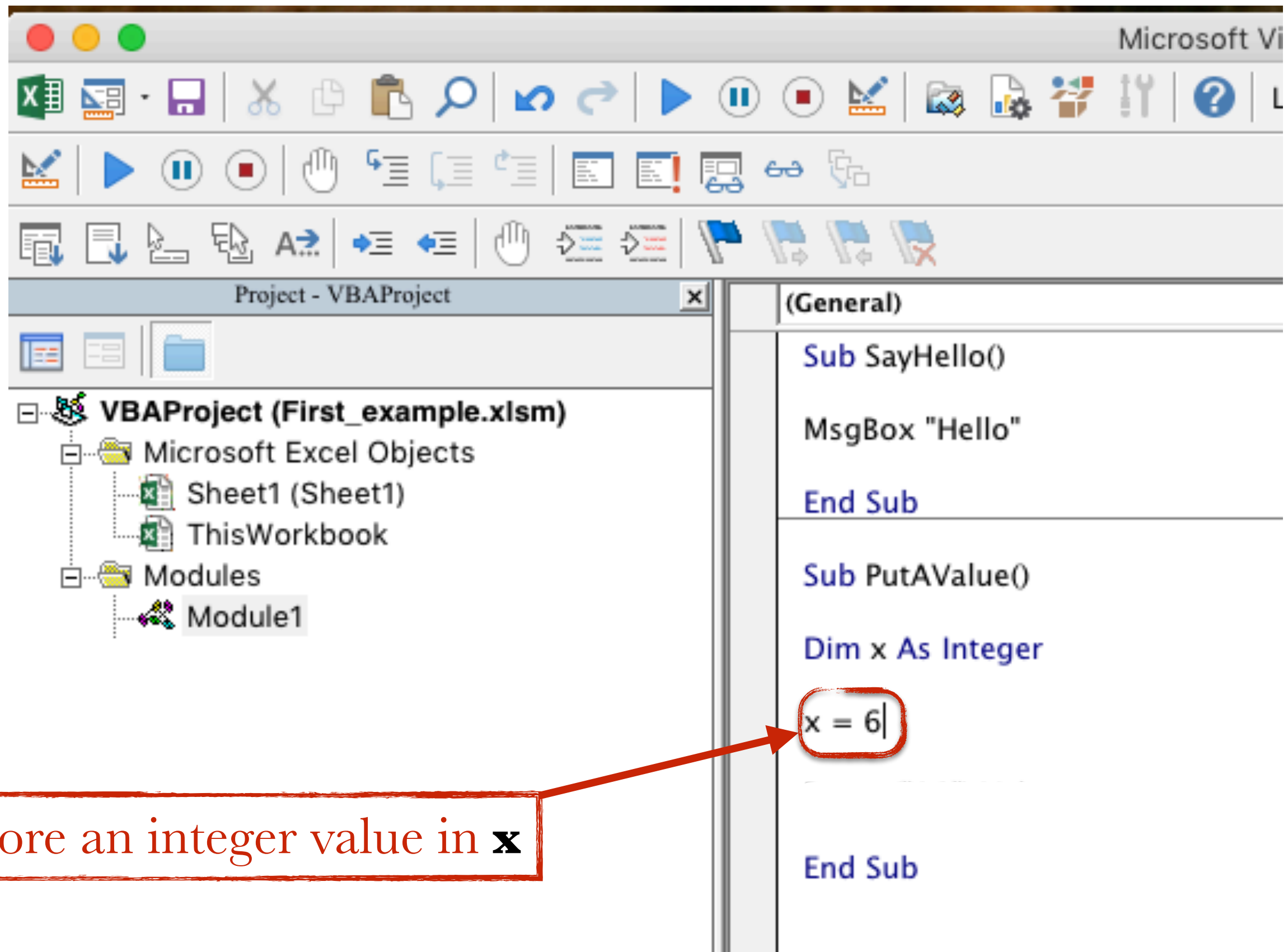
This could be ok for very simple and short codes.

But it could be a serious bug for complex codes!

Efficiency would be seriously compromised!







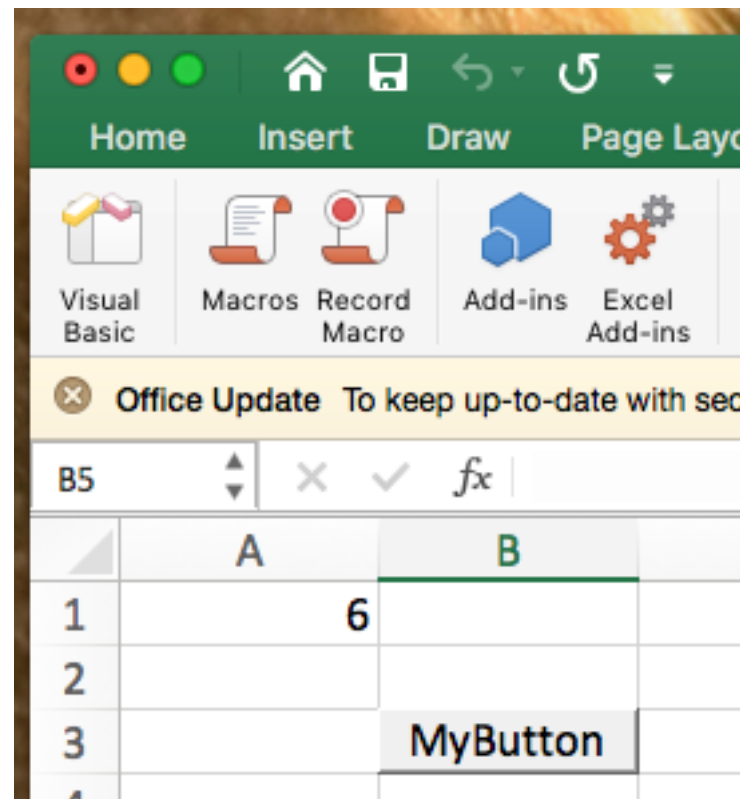
Store an integer value in **x**

The screenshot shows the Microsoft Visual Basic for Applications editor. The left pane displays the Project Explorer for 'Project - VBAProject'. Under 'VBAProject (First_example.xlsm)', there are 'Microsoft Excel Objects' (Sheet1, ThisWorkbook) and 'Modules' (Module1). The right pane shows the code editor for 'Module1' with the following VBA code:

```
Sub SayHello()  
    MsgBox "Hello"  
End Sub  
  
Sub PutAValue()  
    Dim x As Integer  
    x = 6  
    Range("A1").Value = x  
End Sub
```

A red box highlights the line `Range("A1").Value = x` in the `PutAValue()` subroutine. An arrow points from this box to a text box on the left that reads: "Assign the value of **x** to the cell **“A1”**".

Create a new button and attach it the “**PutAValue**” macro



Play with the button!

```
Sub PutAString()
```

```
Dim book As String
```

```
book = "Anna Karenina"
```

```
Range("A1").Value = book
```

```
End Sub|
```

```
Sub PutADouble()
```

```
Dim pi As Double
```

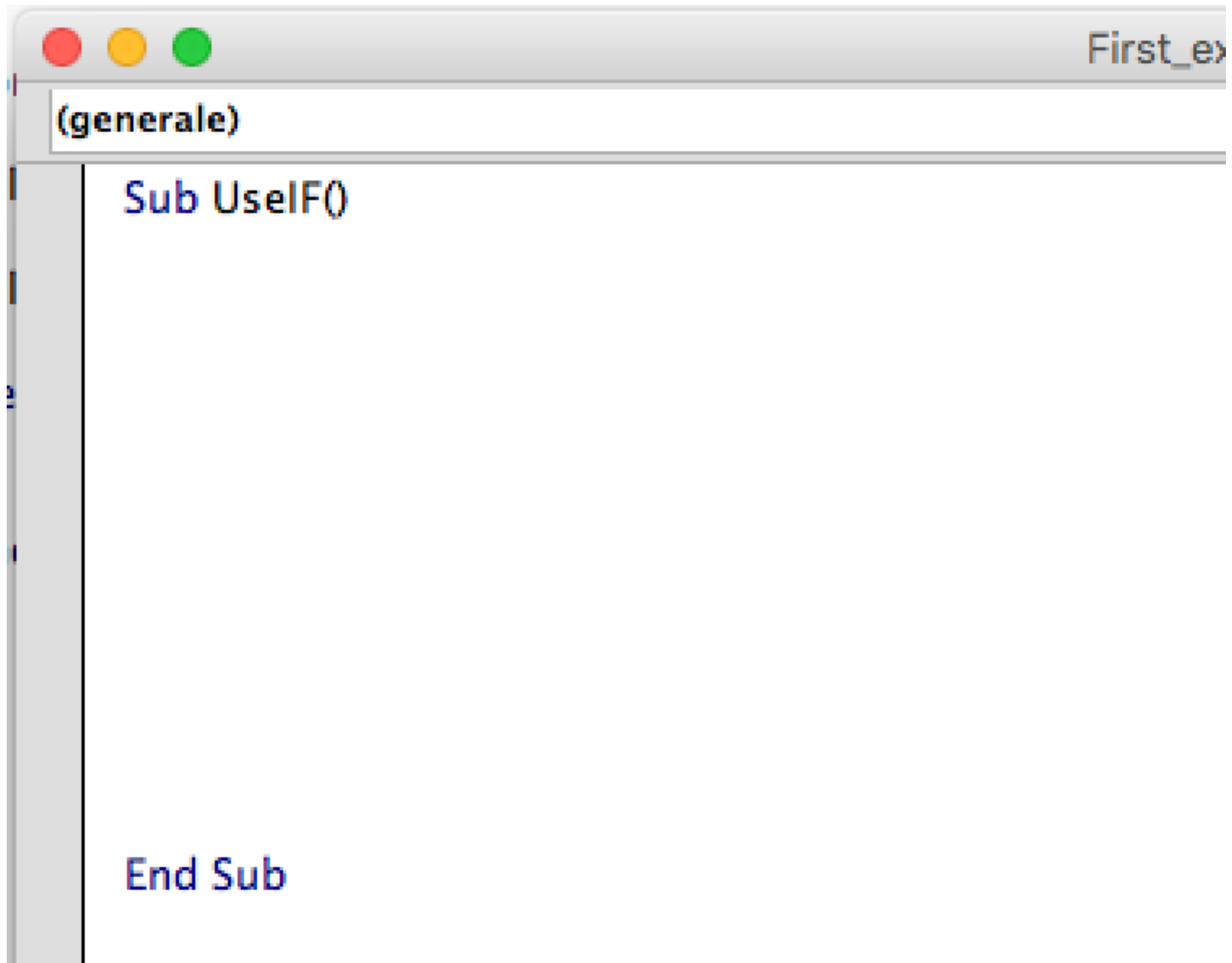
```
pi = 3.14159265359
```

```
Range("A1").Value = pi
```

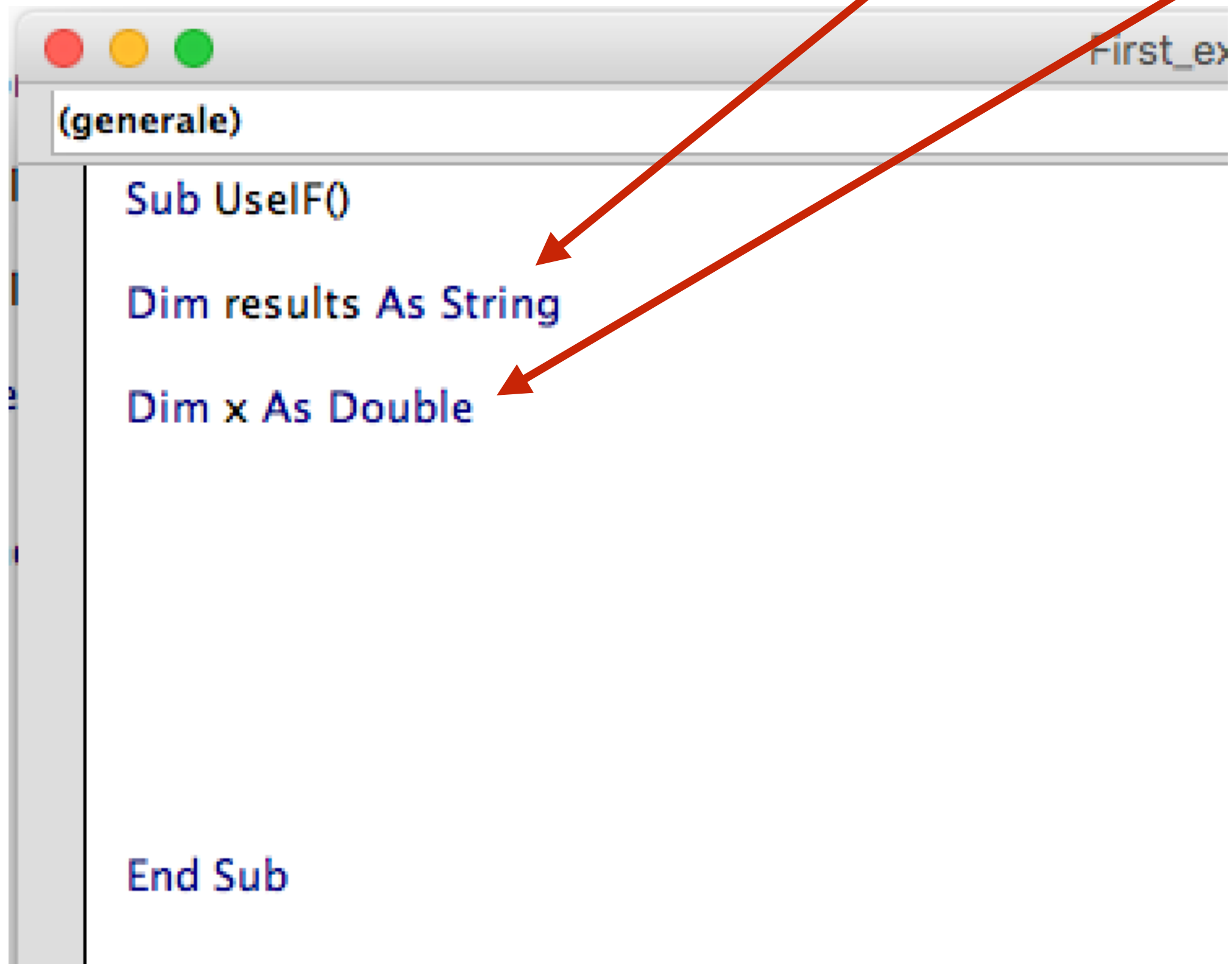
```
End Sub
```

If-else

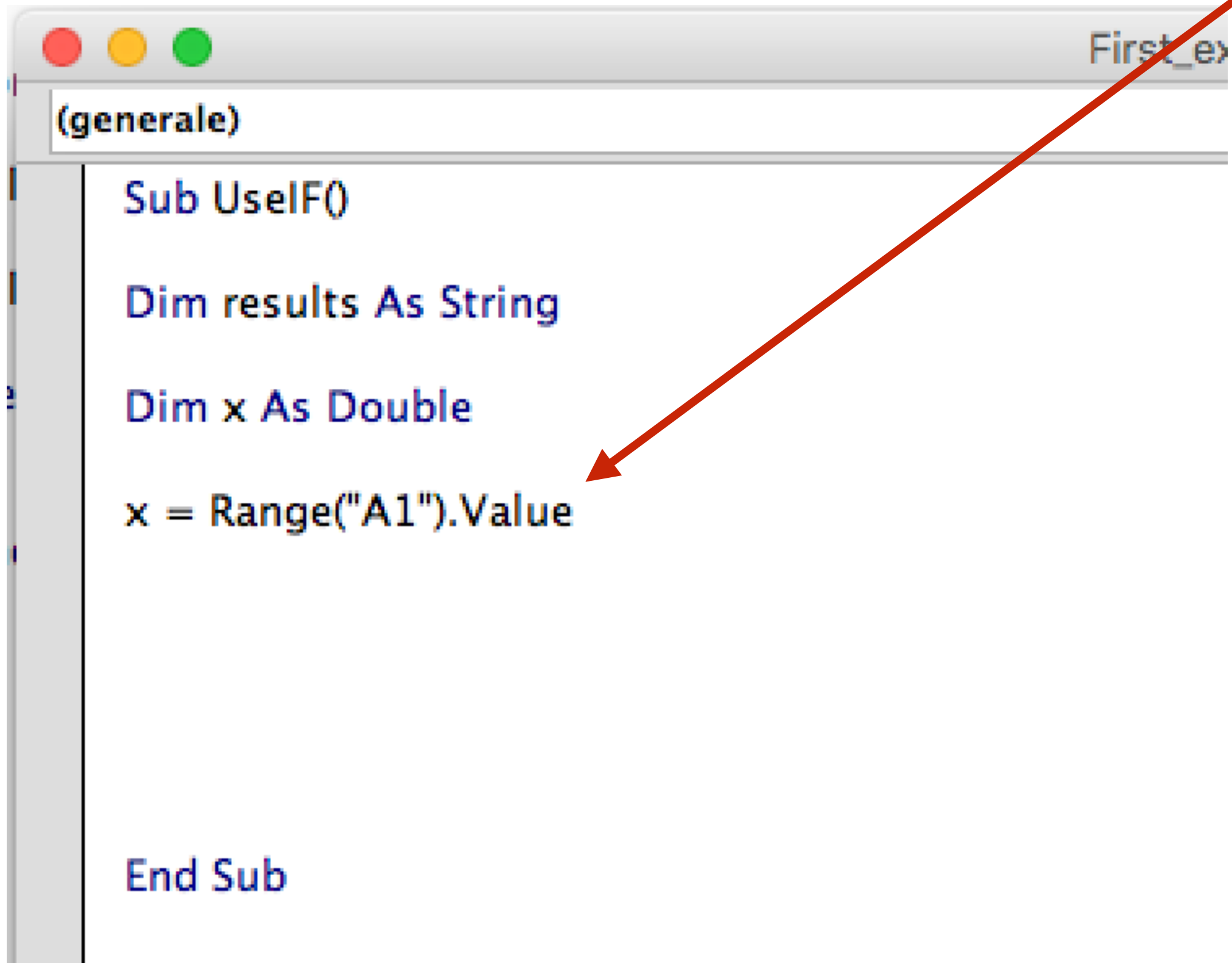
In a new module write:



Variable declarations

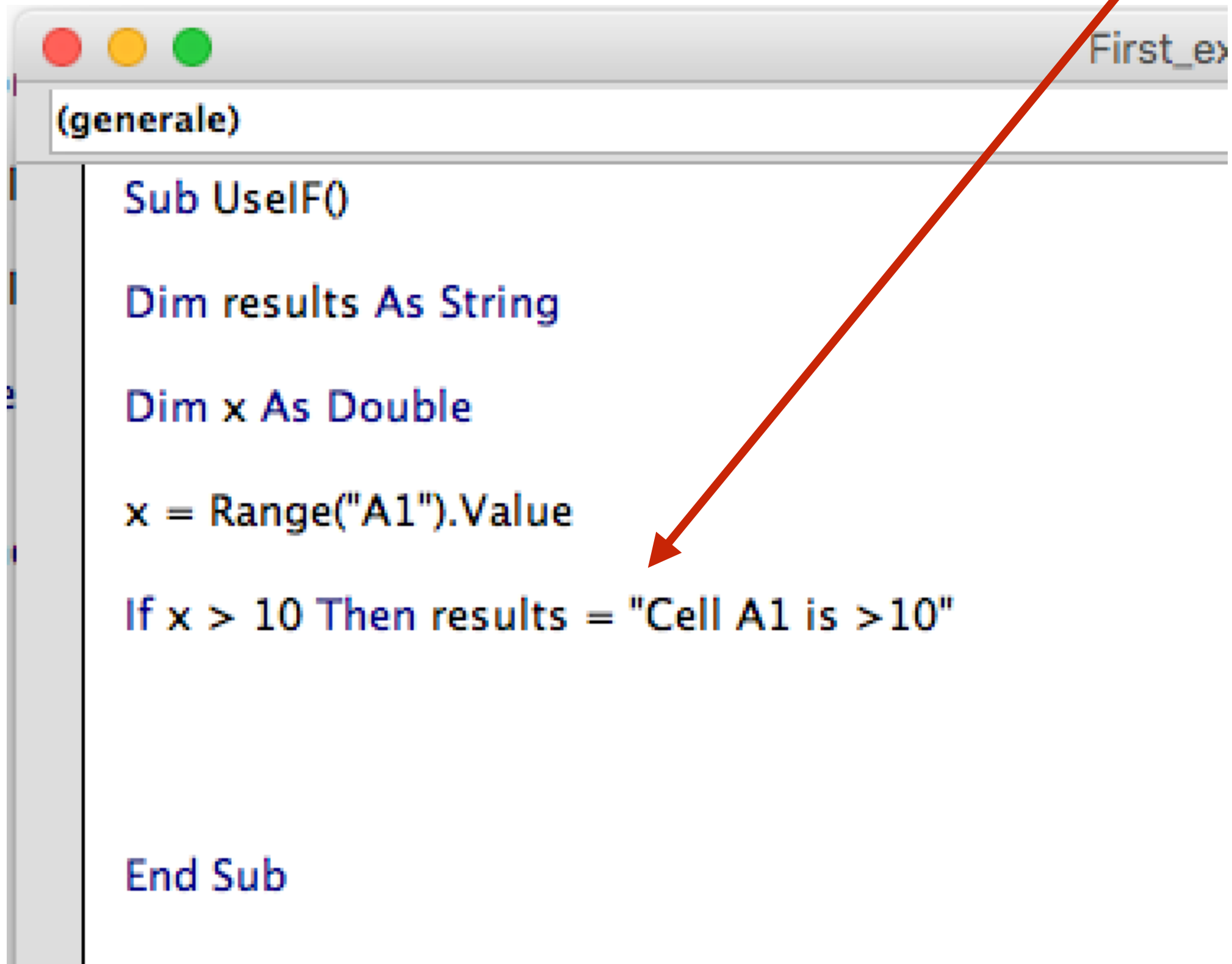


Store in **x** the value
present in the cell "**A1**"



```
(generale)  
  
Sub UseF()  
  
    Dim results As String  
  
    Dim x As Double  
  
    x = Range("A1").Value  
  
  
End Sub
```

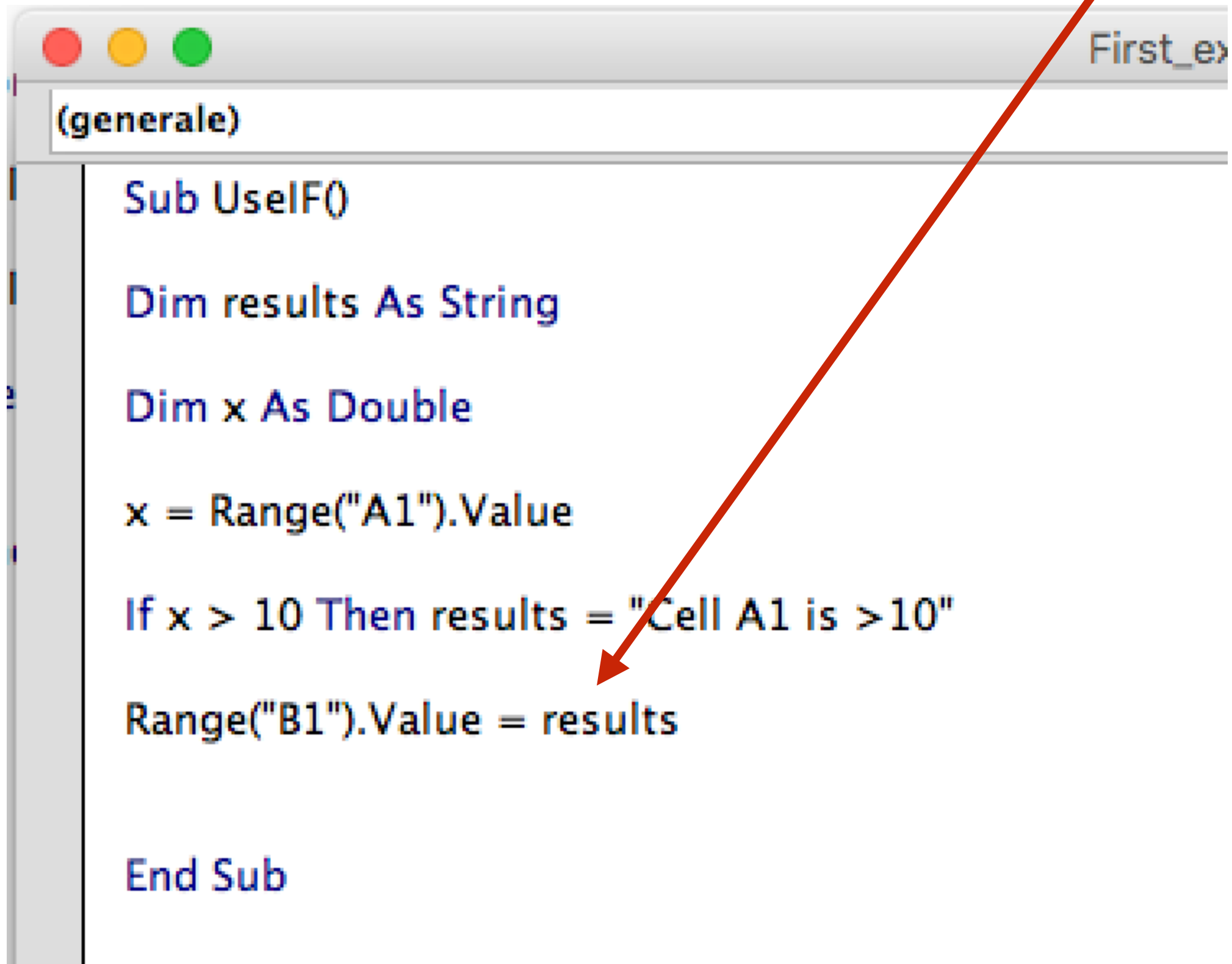
If the value of **x** occurs to be > 10 then store into the variable **results** the string “Cell A1 is >10 ”



```
(generale)
Sub UseIf()
    Dim results As String
    Dim x As Double
    x = Range("A1").Value
    If x > 10 Then results = "Cell A1 is >10"

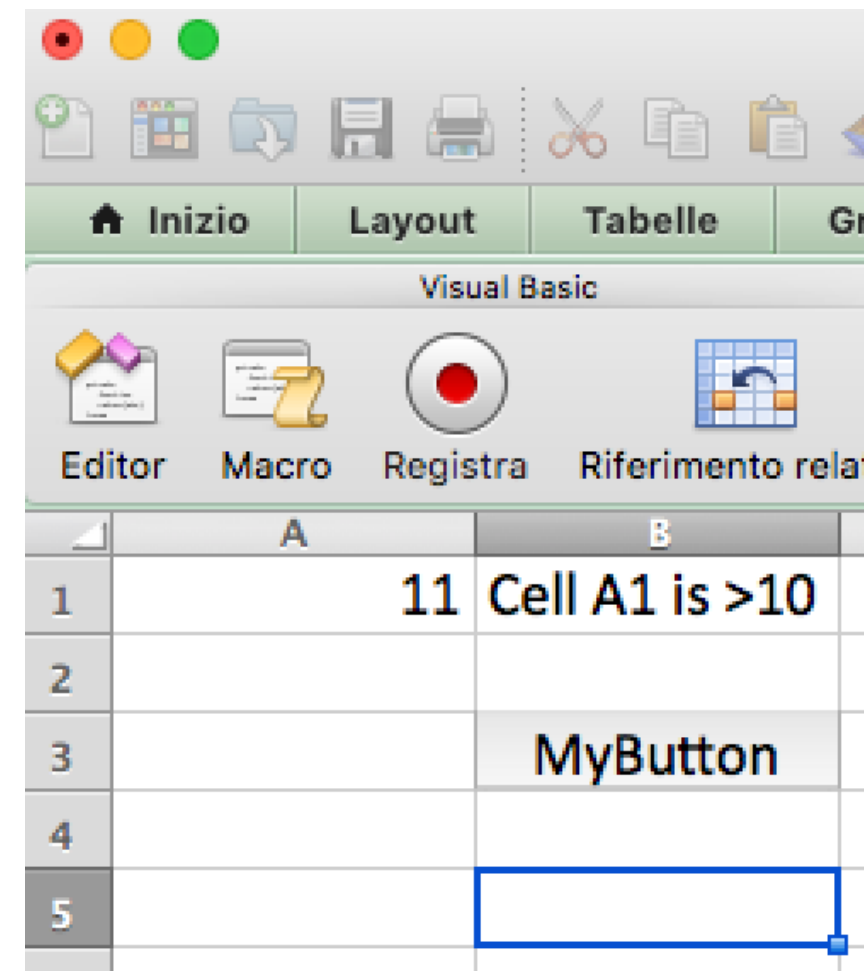
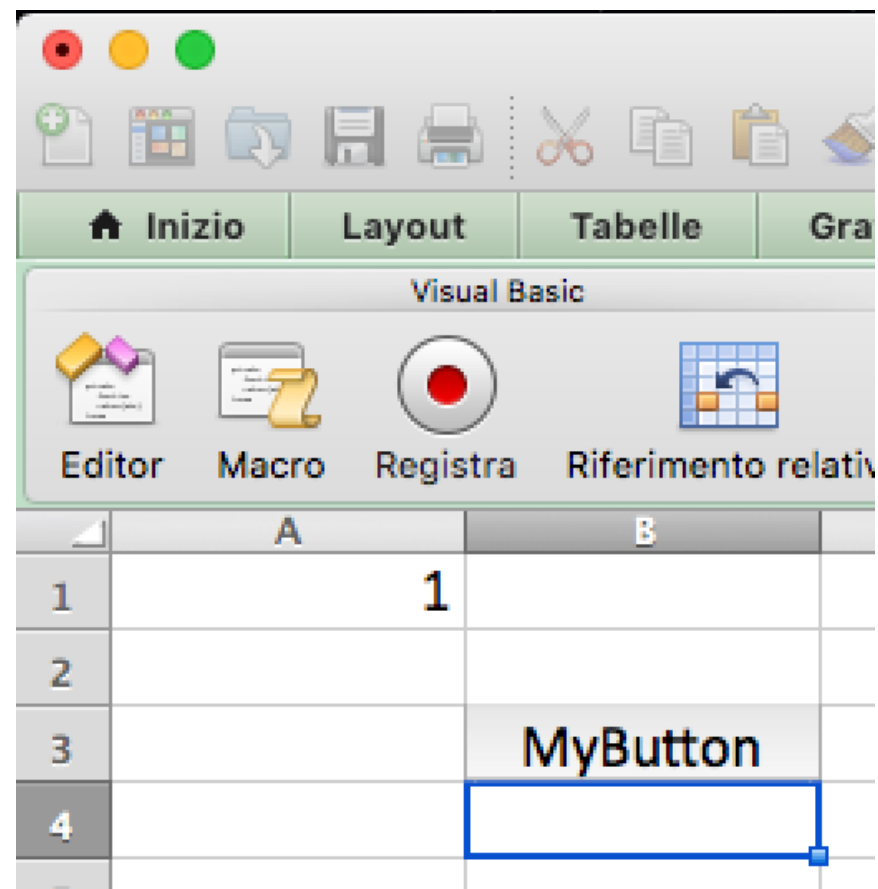
End Sub
```

Assign to the cell “**B1**” the value of **results**



```
Sub UserF()  
  
Dim results As String  
  
Dim x As Double  
  
x = Range("A1").Value  
  
If x > 10 Then results = "Cell A1 is >10"  
  
Range("B1").Value = results  
  
End Sub
```

Create a new button and attach it the “**UseIf**” macro



Play with the button!

In the same module write:

```
Sub UseElse()
```

```
End Sub
```


Compact variable declaration



```
Sub UseElse()
```

```
Dim value As Double, result As String
```

```
End Sub
```

Store into **value** the value of cell **A1**



```
Sub UseElse()
```

```
Dim value As Double, result As String  
value = Range("A1").value
```

```
End Sub
```

If value is >10 execute this



```
Sub UseElse()
```

```
Dim value As Double, result As String  
value = Range("A1").value
```

```
If value > 10 Then
```

```
    result = "Cell A1 contains a number > 10"
```

```
Else
```

```
    result = "Cell A1 contains a number <= 10"
```

```
End If
```

```
End Sub
```

... otherwise execute this.



```
Sub UseElse()
```

```
Dim value As Double, result As String  
value = Range("A1").value
```

```
If value > 10 Then
```

```
    result = "Cell A1 contains a number > 10"
```

```
Else
```

```
    result = "Cell A1 contains a number <= 10"
```

```
End If
```

```
End Sub
```

Whichever is the value of **result** after the **if-else** statement, assign it to the cell “**B1**”

```
Sub UseElse()
```

```
Dim value As Double, result As String  
value = Range("A1").value
```

```
If value > 10 Then
```

```
    result = "Cell A1 contains a number > 10"
```

```
Else
```

```
    result = "Cell A1 contains a number <= 10"
```

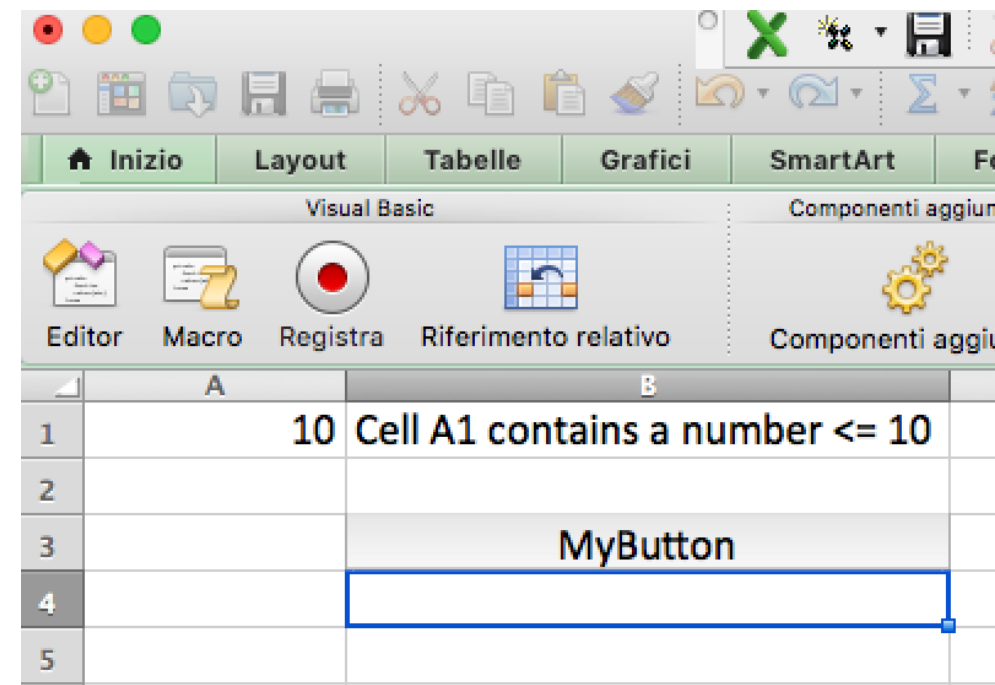
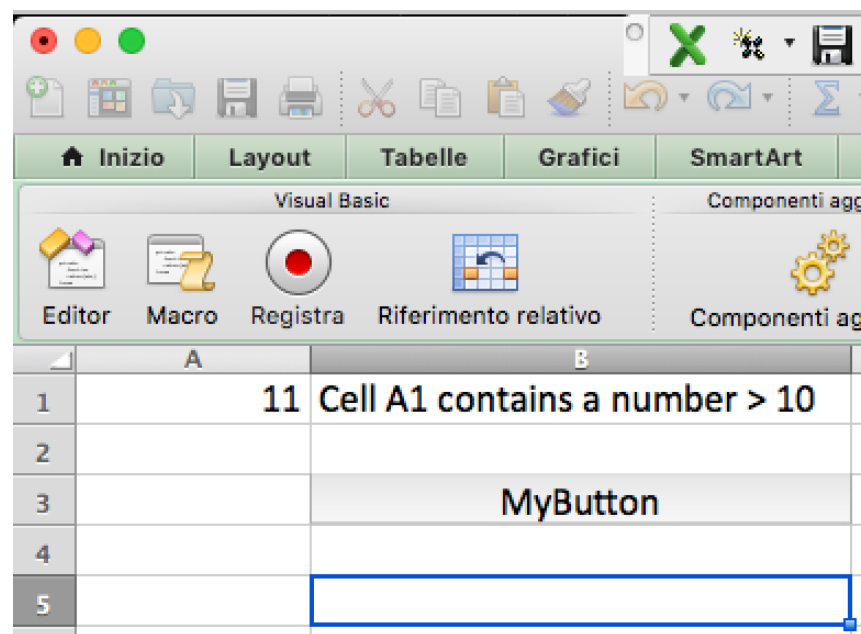
```
End If
```

```
Range("B1").value = result
```

```
End Sub
```



Create a new button and attach it the “**UseElse**” macro



Play with the button!